

JUNE 6, 2021
FACULTY OF TECHNOLOGY, POLICY AND MANAGEMENT
DELFT UNIVERSITY OF TECHNOLOGY
MASTER THESIS COMPLEX SYSTEMS ENGINEERING AND MANAGEMENT

Transparent Decision Support in ever-changing healthcare contexts.

- Designing an architecture of a transparent and dynamic Clinical Decision
Support System grounded in Discrete Choice Modeling -

by

Verena Schrama

Student number: 4473787

Page Count: 115

Defence: June 14th 2021

Graduation committee

First Supervisor: Mark de Reuver, Information and Communication Technology

Second Supervisor: Rens Kortmann, Policy Analysis

External Supervisor: Caspar Chorus, Councyl



Preface

In December 2020, I embarked on the journey of writing a master thesis at Delft University of Technology. The journey started with an internship vacancy at the start-up Council, which immediately caught my attention. It offered a combination of the three courses I enjoyed the most (IC Architecture Design, IC Service Design and Statistical Analysis of Choice Behaviour). Moreover, it enables me to be active in the dynamic context of a start-up. Having combined everything that excites me into a research proposal, I saw a flawless project on my horizon. Obviously, performing a design research turned out to be slightly more challenging than expected. Fortunately, I was surrounded with people that guided me so that I was able to make the journey as flawless as possible.

In general, I am everybody thankful that I have been guided by people that have consistently trusted me and have given me the freedom to add my own spices to the project. To start with, I consider myself very lucky to have the opportunity to do research at Council. Council allowed me to work with very intelligent, motivated, warm, and inspiring people. Moreover, this opportunity gave me insight into what I really like to do: ensuring the design of societal technologies aligns with people's needs.

Caspar Chorus, I want to thank you for helping me with finding the right balance and looking at the bigger picture. Designing an artefact that deals with many details made me feel lost quickly. After a talk with you I was always able to reset my focus on the aspects that required this focus. Moreover, I would like to thank Caspar for guiding me through the complex Discrete Choice Modeling (DCM) theories, and more importantly, finding a balance between incorporating complex DCM practices and effective, understandable decision support.

Nicolaas Heyning, I would like to thank you for all the sparring sessions during which I identified some of my most valuable ideas and the willingness to answer my every question. Moreover, you showed me the added value of scheduling in "down time" to get the most creative ideas. Finally I would like to thank Annebel ten Broeke. Having an example of a very successful thesis project at Council was of great value. The walks and substantive discussions provided me with valuable research tips, tricks, and insights.

The guidance I received from the TU Delft has also been a great support during my thesis process. Mark de Reuver, I would like to express my sincere thanks for being both my first supervisor and the chair of my graduation committee. Although I sometimes felt lost in all the design considerations, each meeting with you provided me with the insights and feedback to improve my work and continue with a clear goal in mind. Especially your feedback on the writing process was invaluable. Because the design process was highly iterative, I found reporting the results of the iterative process clearly challenging. When discussing the challenges I encountered with you, you always knew exactly what to ask or say to let me tackle these challenges. Second, I would like to thank Rens Kortmann for the critical notions and feedback. There are two moments I would like to thank you for in particular. First of all, for showing me to how to position design research in the field of design science research so that it is of scientific value. Second, for warning me about the amount of time that is needed to write down your research findings, especially in case of a design research. Nothing was truer than what you told me in December 2020: schedule at least twice as much time in for writing than you would reasonably expect.

Thanks to the support of all people at Council, Mark and Rens, almost six months later, I can surely say it has been a challenging but rewarding experience.

Verena Schrama
June 3rd 2021

Executive summary

Every day, physicians make decisions on medical treatments that directly influence patients' well-being. Clinical decision-making is concerned with information overload, risk of treatment errors, and risk of treatment costs. To deal with complex decisions and to minimize decision errors, physicians show a growing interest in Clinical Decision Support Systems (CDSS). A CDSS is a computerized system that processes relevant information and provides recommendations to assist physicians in decision-making. Ideally, these CDSSs are transparent and dynamic. Transparency is important because the decisions are concerned with patients' well-being. As such, physicians need to know how a CDSS creates its recommendations. Dynamic refers to the capability to incorporate new clinical developments. If a CDSS is not dynamic, it will lose its accuracy in an ever-changing healthcare decision-making context. So far, there are no technologies in place that facilitate the development of transparent and dynamic CDSSs.

Recently a new technology has been developed: Behavioral Artificial Intelligence Technology (BAIT). BAIT applies Discrete Choice Modeling (DCM) to codify the domain expertise of physicians. In theory, this technological approach allows for the development of CDSSs that are both transparent and dynamic. So far, BAIT has been successfully applied for the development of transparent CDSSs. These CDSSs are referred to as BAIT-based CDSSs. The next step is to apply BAIT for the development of a dynamic CDSSs. These CDSSs are referred to as dynamic BAIT-based CDSSs. However, the healthcare decision-making contexts in which CDSSs are to be applied widely vary. As a result, there is no one-size-fits-all solution for the transformation of a BAIT-based CDSS into a dynamic one. Therefore, CDSS developers need a dynamic BAIT-based CDSS architecture that shows a developer how to build a customized dynamic BAIT-based CDSS that satisfies the needs of physicians in a particular decision-making context. A CDSS architecture is a conceptual model that defines a CDSS's structure and behavior.

The design of a dynamic BAIT-based CDSS architecture is challenging for four reasons. First, the BAIT approach is a novel innovation. The technical and social risks of this approach in dynamic healthcare decision-making contexts are still unknown. Moreover, designing such an architecture involves combining theories and practices from Discrete Choice Modeling (DCM), CDSS design, and Machine Learning. Design studies did not combine these areas before. Third, there are no building blocks in place that facilitate the start of the architecture design. Finally, designing the architecture requires incorporating the preferences of physicians in varying healthcare contexts. Because these contexts are dynamic, physicians' preferences regarding a dynamic BAIT-based CDSS will also change over time.

To expedite the design of dynamic BAIT-based CDSS architectures, this research aims to formulate and test design principles that guide architecture designers in tackling the four above-mentioned challenges. Design principles define the guidelines designers should respect to increase the chance of reaching a successful architecture. To identify the design principles, this research follows a case study in line with the Action Design Research (ADR) framework. The ADR framework makes it possible to identify the requirements of a dynamic BAIT-based CDSS architecture and to test these requirements by building an architecture in a situated problem context. Moreover, trying out different design alternatives in a situated context generates lessons on what works and what does not work when designing a dynamic BAIT-based CDSS architecture. These lessons informed the generalization of the architecture requirements into ten design principles. These principles guide architecture designers outside the situated context in designing a dynamic BAIT-based architecture and in avoiding the obstacles encountered during this research process. Although some design principles confirm the relevance of existing CDSS architecture design principles in the context of a BAIT-based CDSS, the greater part of the requirements is specific to the design of a dynamic BAIT-based CDSS architecture. By providing these novel insights, the design principles contribute

to the CDSS architecture design knowledge base.

Besides contributing to this knowledge base, the design principles contribute by expressing recommendations for CDSS providers. A CDSS provider should contract an architecture designer and a CDSS developer who both possess DCM knowledge and skill. The provider should encourage the architecture designer to design all organizational processes, software processes, and information objects necessary for a dynamic BAIT-based CDSS. Moreover, the designer should create an architecture with an adaptable structure. An adaptable architecture has two advantages.

The first advantage is that an adaptable architecture allows an architecture designer to adjust the architecture towards newly discovered preferences over time. The elicitation of preferences over time requires the CDSS provider to explicitly assign the architecture designer with the role of architecture owner. The owner takes care of the maintenance of the architecture and provides CDSS developers with the architecture guidelines, so CDSS developers will use the architecture as intended. The second advantage is that an adaptable architecture allows CDSS developers to customize a dynamic BAIT-based CDSS. The architecture should allow physicians to choose from different options regarding the level of updating automation and the choice information incorporated in the updates of a CDSS.

The customization that this adaptable approach facilitates might give rise to unintended subjectivity in the updates of a CDSS. Therefore, the provider should instruct the architecture designer to maximize the objectivity with which a CDSS processes novel clinical information during an update. Moreover, the CDSS provider should make physicians aware of the effects of the different options from which physicians can choose. By doing so, the provider ensures that physicians choose specific CDSS features deliberately and the CDSS customization is intended.

The preferences of physicians do not vary regarding all CDSS features of a dynamic BAIT-based CDSS. Some features are essential, independent of the preferences of physicians. Therefore, the CDSS provider should always instruct the architecture designer to pay attention to the following three key CDSS features:

1. The architecture should ensure the CDSS makes transparent how it processes contextual changes during an update. Therefore, the architecture should only incorporate components and outcomes that are comprehensible for clinical end users and that make the change in a CDSS resulting from updates tractable for clinical end users.
2. The architecture should ensure a dynamic BAIT-based CDSS works in partnership with physicians and allows physicians to control the change in a CDSS resulting from updates. The latter ensures that the CDSS bases its recommendations on information physicians find relevant.
3. The architecture should describe a performance assessment component that gives physicians insight into the validity of the CDSS's choice recommendations for the present decision-making context.

While designing these CDSS features, the designer must pay attention to two vital aspects for a viable CDSS. The first vital aspect is that the architecture should not include information flows that a CDSS provider or physicians can relate to individual physicians or that physicians specified as undesired. To this end, the CDSS provider should create an Information Processing Agreement that captures the agreements on information sharing between a CDSS provider and physicians before implementing a dynamic BAIT-based CDSS. The second vital aspect is that the architecture should not include interaction flows that may be unavailable or concern information from which a physician does not benefit. The provider is recommended to validate the architecture design on these two vital aspects.

Finally, the CDSS provider will need to execute new organizational tasks that are vital for the successful implementation of a dynamic BAIT-based CDSS. The CDSS provider should ensure that the architecture designer describes all organizational activities a CDSS provider needs to perform to offer a successful decision support service. Accordingly, the provider should fit these new organizational activities into the existing workflow.

To finish, this research contributes to CDSS architecture design knowledge because it tackles the challenges of designing a dynamic BAIT-based CDSS architecture. By doing so, the research outcomes eliminate the barriers to design such an architecture and lay a foundation to continue the work on transparent and dynamic CDSSs. The utilization of this foundation could start with

the recommendations for further research identified along with this research. The research recommendations fall into three categories. The first category contains research efforts to improve the architecture design, like investigating how clinical choice outcomes can enrich the information a CDSS incorporates during an update. The second category involves researching new design challenges, like designing a dynamic BAIT-based CDSS architecture with the ten design principles in another sector. By doing so, the designer can investigate how to modify the design principles, so the principles are useful outside the healthcare sector. The third category consists of research that aims to fill in the identified knowledge gaps. An example is investigating how to customize the ADR framework so that it better assists designers in tackling the challenges central to architecture design research.

List of Figures

1.1	The construction of a BAIT-based CDSS.	4
1.2	Two choice scenarios for a choice experiment on ICU uptake.	4
1.3	Overview of the stakeholders.	9
1.4	The phases of the Action Design Research (ADR) framework.	12
1.5	Research flow with the input, phases, output, Action Design Research (ADR) principles, and sub-questions.	13
1.6	Design Cycle of	14
2.1	Conceptual overview of the intelligent internal software components of a dynamic CDSS.	20
2.2	Human-Computer Interaction components of a dynamic CDSS.	21
2.3	Conceptual overview of the main components of a dynamic CDSS.	22
2.4	Adaptive Choice Base in the context of a dynamic BAIT-based CDSS.	23
2.5	Model Update Engine in the context of a dynamic BAIT-based CDSS.	27
2.6	Model Quality Monitor in the context of a dynamic BAIT-based CDSS.	27
2.7	Conceptual outline of the design space of a dynamic BAIT-based CDSS architecture.	28
3.1	Stakeholders of the action context: Councilyl and the end users.	31
3.2	Classification of architecture requirements.	33
4.1	Conceptual overview of possible choice influence and weighting extensions.	51
4.2	Detailed overview of possible choice influence and weighting options.	52
4.3	Colored overview of possible choice influence and weighting options that marks extensions.	52
4.4	Conceptual overview of the three Information specification Extensions.	53
4.5	The interface for a clinical end user to enter choice details and request a choice recommendation in the static BAIT-based CDSS.	56
4.6	The interface for a clinical end user to enter choice specifications and request a choice recommendation in a dynamic BAIT-based CDSS.	57
5.1	High overview of the dynamic BAIT-based CDSS architecture for Councilyl with six components.	61
5.2	Architecture solution: Choice recommendation generator and relationships.	61
5.3	Architecture solution: Adaptive choice base module and relationships.	62
5.4	Architecture solution: Model quality monitor module and relationships.	63
5.5	Architecture solution: Model update engine and relationships.	64
5.6	Architecture solution: Information specification Extensions 1, 2 and 3, and relationships (Exp. = experiment choices, RL=real-life choices).	65
5.7	Architecture solution: User settings component and relationships.	66
5.8	Architecture solution: Model management component and relationships.	67
8.1	Design principle 4 and design principle 5 both enhance a CDSS's user involvement.	101
8.2	Design principle 5 relates to design principle 3 and design principle 1.	102
8.3	Design principle 6 supports design principles 1, 7 and 9.	102
8.4	Design principle 9 restricts design principle 7.	102
D.1	The architecture of a BAIT-based CDSS used to serve clinical end users by Councilyl.	144
E.1	The architecture of a dynamic BAIT-based CDSS for Councilyl.	146
E.2	The Information specification Extensions that are part of the architecture of a dynamic BAIT-based CDSS for Councilyl.	147

F.1	High overview of the dynamic BAIT-based CDSS architecture for Councilyl with six components.	149
G.1	The python packages needed for the proof-of-technology. Required for implementation of all components in table G.1.	151
G.2	The code to retrieve and prepare the choice data in the adaptive choice base for an update. Required for implementation of all components, but represents component 1 in table G.1.	152
G.3	The code to estimate and validate a recommendation generator model for an update with the choices stored in in the temporary choice bases. Required for implementation of all components in table G.1.	153
G.4	The code to adjust weight with which experiment choices (SP) and real-life (RP) choices are multiplied (front-end) and to store the multiplied choices in the temporary choice base (back-end). This code shows how an user can specify the weight with an importance rate (a multiplication of choices with a chosen integer). Required for implementation of component 2 in table G.1.	154
G.5	The code to adjust weight with which experiment (SP) and real-life (RP) choices are multiplied (front-end) and to store the multiplied choices in the temporary choice base (back-end). This code shows how an user can specify the weight with an importance balance (a multiplication of choices with a balance slider). Required for implementation of component 2 in table G.1.	154
G.6	The code to select a sub group of choices (for instance all choices made by juniors) and exclude these choices or multiply these choices with a weight (here an importance rate, alternative is importance balance). Required for implementation of component 3 and 4 in table G.1.	155
G.7	The code to create the temporary choice bases for an update according to the specifications of the clinical end user (see fig. G.6). Required for implementation of components 3 and 4 in table G.1.	155
H.1	Performance metrics overview for Councilyl.	159
H.2	Access to user settings for Councilyl to make specific sub groups unavailable according to the Information Processing Agreement for privacy reasons.	160
H.3	The login screen for both Councilyl and clinical end users.	161
H.4	The real-life choice storage and recommendation request screen. A clinical end user can enter the specification of a choice task he or she has to deals with. While filling in the choice specifications, the choice recommendation generator dynamically determines the recommendation. The end user can hide the recommendation if he or she does not want to be influenced in filling the choice details or in making a choice. Finally, the real-life choice specification can be stored.	161
H.5	A brief confirmation of a dynamic BAIT-based CDSS that the choice stored.	162
H.6	The model insight screen shows the active model (in terms of the relative importance of all choice attributes for the decision-making) and the Correspondence rate the choice/clinical end user agreement table that were determined during the model validation process. Moreover, end users can request previous model updates to compare different model updates. Finally, it can be observed how the importance of the choice attributes changes over time (per update).	162
H.7	The decision investigation allows clinical end users to specify a sub group of choices for which they want to see the model estimation and validation. For instance, the model for all choices made by senior clinical end users can be requested. The goal of this tool is either mutual learning or trying out model updates with different sets of choices.	163
H.8	As soon as a clinical end user has requested the model for a specific sub group, the model is directly and locally presented.	163
H.9	The screen where clinical end users can activate a model update. The CDSS will directly update with the choices as specified in the user settings.	164
H.10	The CDSS confirms an update, but does not directly presents the results of the update.	164
H.11	The screen where clinical end users can reset the model that is used by the choice recommendation generator to provide choice recommendation. To this end, the end user can choose from all previous model updates.	164

H.12	The CDSS briefly confirms the selection of a new model for the choice recommendation generator.	165
H.13	A clinical end user can request all experiment choices and real-life choices he or she made in the past.	165
H.14	The end user can access all the choices for which he or she deviated from the recommendation of the dynamic BAIT-based CDSS.	166
H.15	The notification screen gives an overview of all warning feedback that is provided by the CDSS. The notification that warns that an update is needed remains visible until the end user updates the model.	166
H.16	The user setting for the preference values show all options from which end users can choice to customize the CDSS towards the particular preferences. Most importantly, a clinical end user can choose here for the desired level of updating automation. . .	167
H.17	One or more of these screens are only accessible if the clinical end user chose Information specification Extension 1, 2 or 3. If implemented, these screens allow clinical end users to specify how much specific types of choices will influence the model updates. To specify the weight, the end user can choose between an importance rate or an importance balance. All settings can be tried out, meaning that the results will be presented but will not be stored or influence the model that the CDSS uses for the generation of choice recommendation.	168
J.1	Data overview architecture of a dynamic BAIT-based CDSS for Councyl.	172

List of Tables

1	<i>Acronyms and shortcut words.</i>	
	.25table.caption.33	
2.2	<i>The architecture design space framework.</i>	29
3.1	<i>Definitions of the architecture requirement categories</i>	33
3.2	<i>Questions and rationale semi-structured interviews Councilyl</i>	36
3.3	<i>Questions and rationale semi-structured interviews clinical end users</i>	37
3.4	<i>Client Artefact Requirements</i>	39
3.5	<i>Quality Attribute Requirements</i>	41
3.6	<i>Development Guiding Requirements</i>	45
4.1	<i>The features of real-life choices.</i>	51
6.1	<i>Evaluation of usefulness</i>	71
6.2	<i>Evaluation of understandability</i>	72
6.3	<i>Overview table of the implemented and tested architecture components</i>	74
8.1	<i>Overview of the design principles for designing a dynamic BAIT-based CDSS architecture.</i>	91
8.2	<i>Architecture Requirements design principle 1: Adaptable design.</i>	92
8.3	<i>Architecture Requirements design principle 2: Objectivity maximization within the subjective boundaries.</i>	94
8.4	<i>Architecture Requirements design principle 3: Goal-based interaction.</i>	95
8.5	<i>Architecture Requirements design principle 4: Tractability of change.</i>	96
8.6	<i>Architecture Requirements design principle 5: Receptivity to user input.</i>	97
8.7	<i>Architecture Requirements design principle 6: Differentiation in consumed choices and produced information.</i>	98
8.8	<i>Architecture Requirements design principle 7: Mutual learning.</i>	99
8.9	<i>Architecture Requirements design principle 8: Architecture intuitiveness.</i>	100
8.10	<i>Architecture Requirements Design principle 9: Privacy of choice information and decision-making behaviour information.</i>	100
8.11	<i>Architecture Requirements design principle 10: Explication of the organizational activities.</i>	101
A.1	<i>List of the requirement identification interviews</i>	129
C.1	<i>Definition of relationships specified ArchiMate.</i>	143
C.2	<i>Additional specifications and shortcut words.</i>	143
G.1	<i>Overview table of the implemented and tested architecture components</i>	150
G.2	<i>Parameters based on experiment choices, and on experiment choices and real-life choices (joint) with component 1 (see appendix G.1)</i>	157
G.3	<i>Correspondence rates for model estimation based on experiment choices and real-life choices</i>	157
G.4	<i>Parameters estimated with component 1 and 2 (see appendix G.1) based on a temporary in which experiment choices and real-life choices were weighted (Exp. = experiment choices, RL=real-life choices).</i>	158

Acronyms and shortcut words

Table 1: *Acronyms and shortcut words.*

Acronym or shortcut	Explanation of meaning in thesis
BAIT	Behavioral Artificial Intelligence Technology
CDSS	Clinical Decision Support System
CDSS architecture	A formal description of all components of a CDSS to guide its development and implementation.
CDSS update	The estimation and activation of a new choice recommendation generator model with new choice information that became available since the previous update.
Choice information	All experiment and real-life choices made by physicians.
Choice recommendation generator model	The Discrete Choice Modeling choice model that a CDSS operates to generate choice recommendations for end users (also referred to as model or choice model).
Clinical end user	A physician who consults a dynamic BAIT-based CDSS for a choice recommendation for assistance on a choice task.
Decision-making behaviour	Physicians' implicit decision-making rules, strategies, and behaviour.
Decision-making behaviour information	An overview of the relative importance clinical end users assign to choice attributes when engaging in a particular decision-making task.
Decision-making context	The space that is made up by all factors that influence the outcome of a particular choice task.
Dynamic CDSS	A Clinical Decision Support System that incorporates new contextual information and, by doing so, remains or even improves the quality of its decision recommendations during ongoing use.
Experiment choice	A choice clinical on treatment for a hypothetical patient explicated during a controlled choice experiment (also referred to as a stated preference (SP)).
IDSS	Intelligent Decision Support System
Introspection	The act of investigating physicians' decision-making behaviour.
KBS	Knowledge-based system
NKBS	Non knowledge-based system
Real-life choice	A choice on clinical treatment for a patient that a physician expressed in a real-life setting (also referred to as revealed preference (RP)).

Contents

Preface

List of figures

Acronyms

1	Introduction	2
1.1	Decision support in the healthcare sector	2
1.2	BAIT approach for a new type of CDSS	3
1.2.1	Introduction to Discrete Choice Modeling	3
1.2.2	Introduction to Behavioral Artificial Intelligence Technology	3
1.3	The dynamic application context of CDSSs	5
1.4	Research objective	6
1.4.1	The value of a BAIT-based CDSS	6
1.4.2	The value of architecture design	7
1.4.3	Filling the gap of guiding design principles	8
1.4.4	Research questions	9
1.5	Research approach	10
1.5.1	Selection of the research approach	11
1.5.2	Research flow	12
1.6	Summary chapter 1	15
2	Definition of the architecture design space	17
2.1	Architecture components of a CDSS	17
2.1.1	Intelligent internal software	18
2.1.2	Human-Computer Interaction	20
2.2	The design space set by Discrete Choice Modeling	22
2.2.1	Adaptive knowledge base	22
2.2.2	Model update engine	24
2.2.3	Model quality monitor	27
2.3	Summary chapter 2	28
3	Introduction to the action context: The current and desired situation	30
3.1	Introduction to the current action context	30
3.1.1	Introduction to Council and the key stakeholders	30
3.1.2	Introduction to the problem experienced by Council	31
3.2	Desired situation: Architecture requirements	32
3.2.1	The classification of architecture requirements	32
3.2.2	The methodology of the requirement identification	33
3.2.3	The results of the requirement identification	38
3.3	Summary chapter 3	44
4	Specification of the Architecture: Design Decisions	46
4.1	Design decisions on the architecture structure	46
4.1.1	Layered design	46
4.1.2	Adaptable design	47
4.1.3	Soft- and hardware independence	47
4.2	Design decisions on the estimation and validation of recommendation generator model updates	48
4.2.1	Extensions involvement and update trigger automation	48

4.2.2	Pooled estimation	48
4.2.3	Restricting the individual parameters with missing values to zero	48
4.2.4	Selection of performance metrics	49
4.2.5	K-fold cross validation with manipulated data split	49
4.3	Design decisions on the customization of the update engine	50
4.3.1	Module extensions for choice inclusion and weight specification	50
4.3.2	Multiplication of sample size to realize the weight specification	53
4.3.3	Temporary choice bases	54
4.3.4	Information specification Extensions as user settings	55
4.4	Design decisions on information management	55
4.4.1	The measurement of new real-life choices over time.	55
4.4.2	Separated choice bases for experiment and real-life choices	55
4.4.3	Data accessibility for clinical end users	56
4.4.4	Split in user settings Council and clinical end users	56
4.4.5	Data accessibility for Council and clinical end users	58
4.5	Summary chapter 4	58
5	Results: Architecture solution	60
5.1	Architecture overview	60
5.1.1	The relation with the current static BAIT-based CDSS	61
5.1.2	Relations between the components of a dynamic BAIT-based CDSS	61
5.2	Adaptive choice base	62
5.3	Model quality monitor module	63
5.4	Model update engine module	64
5.4.1	The construction of temporary choice bases	64
5.4.2	The choice recommendation generator model estimation	65
5.4.3	The choice recommendation generator model validation	65
5.5	User settings component	66
5.6	Model management module	67
5.7	Summary chapter 5	68
6	Evaluation of the architecture solution	69
6.1	Evaluation Approach	69
6.2	Static-oriented evaluation	70
6.3	Dynamic-oriented evaluation	71
6.3.1	Proof-of-technology	72
6.3.2	Mock-ups	74
6.4	Summary chapter 6	76
7	Reflection	77
7.1	The lessons learned: Reflection on the design project	77
7.1.1	Problem framing	77
7.1.2	Emerging Artefact	79
7.1.3	Fundamental theories	84
7.1.4	Design Process	87
7.2	Summary chapter 7	88
8	Generalization	90
8.1	Design principles	90
8.1.1	Design principle 1: Adaptable design	91
8.1.2	Design principle 2: Objectivity maximization within the subjective boundaries	93
8.1.3	Design principle 3: Goal-based interaction	94
8.1.4	Design principle 4: Tractability of change	94
8.1.5	Design principle 5: Receptivity to user input	96
8.1.6	Design principle 6: Differentiation in consumed choices and produced information	97
8.1.7	Design principle 7: Mutual learning	98
8.1.8	Design principle 8: Architecture intuitiveness	99
8.1.9	Design principle 9: Privacy of choice information and decision-making behaviour information	100

8.1.10	Design principle 10: Explication of the organizational activities for the CDSS provider	100
8.2	The relations within the set of design principles	101
8.3	Contribution to architecture design science knowledge	102
8.3.1	Contribution design principle 1: Adaptable design	103
8.3.2	Contribution design principle 2: Objectivity maximization within the subjective boundaries	104
8.3.3	Contribution design principle 3: Goal-based interaction	104
8.3.4	Contribution Design principle 4: Tractability of change	104
8.3.5	Contribution design principle 5: Receptivity to user input	105
8.3.6	Contribution design principle 6: Differentiation in choices	105
8.3.7	Contribution design principle 7: Mutual learning	105
8.3.8	Contribution design principle 8: Architecture intuitiveness	106
8.3.9	Contribution design principle 9: Privacy of choices and decision-making behaviour information	106
8.3.10	Contribution design principle 10: Explication of the organizational activities	107
8.4	Summary chapter 8	107
9	Conclusion	109
9.1	Main findings: design principles for a dynamic BAIT-based CDSS architecture	109
9.2	Theoretical contributions of the design research	110
9.2.1	Design principles	110
9.2.2	Foundation for BAIT-based CDSS architecture design work	111
9.2.3	Reusable solution concepts	112
9.2.4	Contribution to Machine Learning knowledge field	112
9.3	Limitations and Recommendations on further research	113
9.3.1	Limitations of the design principles	113
9.3.2	Recommendations for further research	114
	References	116
A	Overview of the interviews	129
B	Interview analysis	130
B.1	Interviews Council	130
B.1.1	Interview part 1: Main purpose	130
B.1.2	Interview part 2: Functions of the architecture	132
B.1.3	Interview part 3: Characteristics of the architecture	135
B.2	Interviews clinical end users	136
B.2.1	Interview part 1: The design of the updating	136
B.2.2	Interview part 2: Frequency and activation of updating	139
B.2.3	Deviating choices	139
B.2.4	Information provision on the model performance	141
B.2.5	Additional decision-making behaviour insights	141
B.2.6	Majority threshold	142
C	Archimate legend and relationship description	143
D	Architecture current situation	144
E	Architecture solution for Council: full overview	145
F	Architecture solution for Council: high-level overview	148
G	Architecture proof-of-technology: scripts and outcomes	150
G.1	Scripts proof-of-technology	150
G.2	Outcomes of executing scripts	156
H	Architecture implementation: mock-ups	159
H.1	Mock-ups point of view from Council	159
H.2	Mock-ups point of view from clinical end users	161

I	Organizational architecture implementation guidelines for Council	169
J	Overview of the data objects defined by the architecture of a dynamic BAIT-based CDSS for Council	171

Chapter 1

Introduction

1.1 Decision support in the healthcare sector

Every day, physicians make decisions on clinical treatments that directly influence patients' well-being. Decision-making is the practice of choosing between feasible options to select the best alternative (Chikwe, n.d.; Delen, 2019). For physicians, decision-making is concerned with information overload, risk of treatment errors, and risk of treatment costs (Varonen, Kortteisto, Kaila, & Group, 2008). Moreover, the decisions directly influence patients' well-being. Despite that physicians are experts who exhibit powerful knowledge and experience, the decisions they face still form recurring risky challenges with direct impact on the healthcare organization and society (Kirkeboen, 2009).

To deal with such complex decisions and minimize decision errors, physicians show a growing interest in Clinical Decision Support Systems (CDSSs) (Sutton et al., 2020). Since their introduction in the 1980s, CDSSs have seen a rapid evolution (Sutton et al., 2020). CDSSs form a specific subcategory of Decision Support Systems (DSS's) that are applied in the healthcare sector. A DSS is a computerized system that processes and provides relevant information to assist in decision-making, and by doing so, enhances human judgement (Bhatt & Zaveri, 2002; K.-W. Lee & Huh, 2006; Power, 2008; Shim et al., 2002; Zeleznikow & Nolan, 2001). A DSS can be considered as a system providing a set of functions that extend cognitive decision-maker abilities of the end users (Zachary, 1988). An example of a DSS applied in the healthcare sector is a DSS providing recommendations to a physician in favour or against surgery, given a patient's specific conditions. CDSSs thus function as direct aid to the complexity of clinical decisions by interpreting individual patient data and accordingly provide a recommendation regarding the patient (Sutton et al., 2020).

CDSS developers create the CDSS's for physicians. These developers may work for a CDSS provider organization or be active in the academic world to research CDSS's. CDSS developers have been studying and improving DSSs since the 1960s (Behmel, Damour, Ludwig, & Rodriguez, 2021). This ongoing work lead to the development of Intelligent Decision Support Systems (IDSSs) (Behmel et al., 2021). IDSSs are based on artificial intelligence (AI) and aim to support decision-making by mimicking human capabilities and predicting choice recommendations for a physician's choice task (Yilmaz & Tolk, 2008). A prediction entails utilizing available information to generate information that is absent. By doing so, an IDSS can extrapolate insights that can help facilitate decision-making using existing data. The health sector was one of the first domains applying an IDSS (Gulavani & Kulkarni, 2014). Because current CDSSs mainly operate AI techniques, the notion of CDSS refers to an IDSS - a CDSS that is based on AI - in the remainder of this research. Nowadays two main types of CDSSs exist in the literature: non-knowledge-based and knowledge-based systems (Abbasi & Kashiyarndi, 2010). They mainly differ in their technological foundation.

Non-knowledge-based systems (NKBS). NKBS's apply the rapidly growing branch of AI known as machine learning (ML) (Abbasi & Kashiyarndi, 2010; Montani & Striani, 2019). They ground their decision-support on feature extraction of labelled training data to recognize and analyse patterns from unseen data with the use of ML techniques such as deep learning or super vector machines (Burrell, 2016). With the use of ML techniques, there is no need for input of experts and no necessity to write rules as the input like knowledge-based systems require. ML allows CDSSs to learn by including a feedback loop that re-uses the predicted outputs to train new versions of the model (Jordan & Mitchell, 2015).

Knowledge-based systems (KBS). KBS's give recommendations based on domain knowledge in if-then rules (Abbasi & Kashiyarndi, 2010; Montani & Striani, 2019). KBS's were developed early on in the field of AI and uses domain knowledge as a frame of reference. More precisely, it directly translates domain knowledge of experts into a set of rules or cases to support human decisions. As such, they heavily rely on human expertise. Therefore, a KBS is also referred to as an expert system.

1.2 BAIT approach for a new type of CDSS

Recently a third type of CDSS has been introduced: a CDSS that is based on Behavioral Artificial Intelligence Technology (BAIT). A BAIT-based CDSS applies Discrete Choice Modeling (DCM) to codify the domain expertise of physicians (Ten Broeke, Hulscher, Heyning, Kooi, & Chorus, 2021). This section gives an introduction to DCM subsection 1.2.1 and explains how DCM is applied by BAIT subsection 1.2.2.

1.2.1 Introduction to Discrete Choice Modeling

Discrete Choice Modeling (DCM) is used to analyse decision-making and predict future choices of individuals (J. J. Louviere, Flynn, & Carson, 2010). In the analysis of DCM, the preferences of an individual are considered as a set of parameters evaluating how the individual values the attributes in his or her choice-making process (Zhu, Feng, Huang, & Chen, 2020). DCM enriches decision-making researchers or econometricians with two scientifically valuable practices.

First, researchers can estimate a choice model. During the model estimation, a researcher infers the weights that decision-makers attach to different choice attributes (Bech, 2003; Greene, 2009). In this report, the decision-makers are physicians choosing to proceed with a particular clinical operation or not. Choice attributes represent the main criteria that a physician considers when making a choice on a particular clinical operation (Bech, 2003; J. Louviere & Timmermans, 1990). Examples of choice attributes are the BMI of a patient or the Intensive Care Unit (ICU) capacity when deciding on the uptake of a patient in the ICU. The weights represent how important a physician finds a choice attribute relative to the other choice attributes when making a choice. As such, the weights contain crucial information on the likelihood that a physician will choose to operate. With the knowledge of what an individual physician finds important, a modeller can predict whether a physician is likely to choose a particular treatment option for future choice tasks (Train, 2009; Zhu et al., 2020). By inferring the weights physicians attach to choice attributes, a choice model makes the trade-offs of physicians between these attributes explicit. The power of DCM is that the model reveals these trade-offs indirectly instead of asking an individual for trade-offs directly. Decision-makers often don't know what they find more important or hesitate to give the true answers in more sensitive situations. Moreover, judgment is more susceptible to bias than choices in which decision-makers make the trade-offs.

Second, DCM allows the researcher to explore which decision-making rule best describes how the physicians made decisions regarding patients. Different models following a specific decision-making rule exists. The BAIT approach is not bound to a single choice model formulation. To give an illustration, two commonly acknowledged choice models are Random Utility Maximization (RUM) and Random Regret Minimization (RRM). The RUM model assumes that individuals make choices amongst a finite set of available alternatives while aiming to achieve the highest utility. The RRM model assumes individuals make choices amongst a finite set of available alternatives while aiming to avoid the situation where one or more non-chosen alternatives perform better than the one that was chosen to minimise regret after the choice event (Chorus, 2010). Despite the particular decision-making rule assumed, a choice model always consists of an observed and unobserved part. The unobserved part is the error term representing the choice modeller's incapacity to observe all variables that shape an individual's choice. The unobserved part covers measurement errors, differences between individuals, and the randomness inherent in human nature. The variables (choice attributes) that the modeller can measure form the observed part of the choice model (Greene, 2009).

1.2.2 Introduction to Behavioral Artificial Intelligence Technology

A BAIT-based CDSS forms a customized system for a particular recurring choice task. The development of a BAIT-based CDSS consists of two main steps. The first step is conducting a choice experiment (see the left box in Figure 1.1). During the choice experiment, a group of physicians

makes choices on a set of hypothetical choice scenarios. The choices made during this experiment are always hypothetical because a physician only states that he or she would choose a particular alternative if the physician would encounter the scenario (de Freitas, Becker, Zimmermann, & Axhausen, 2019). Because the choices made during the choice experiment reflect the choices a physician states he or she would make, these choices are officially referred to as stated preferences (SP) (M. Ben-Akiva et al., 1994). This research report uses the term experiment choices to refer to stated preferences.

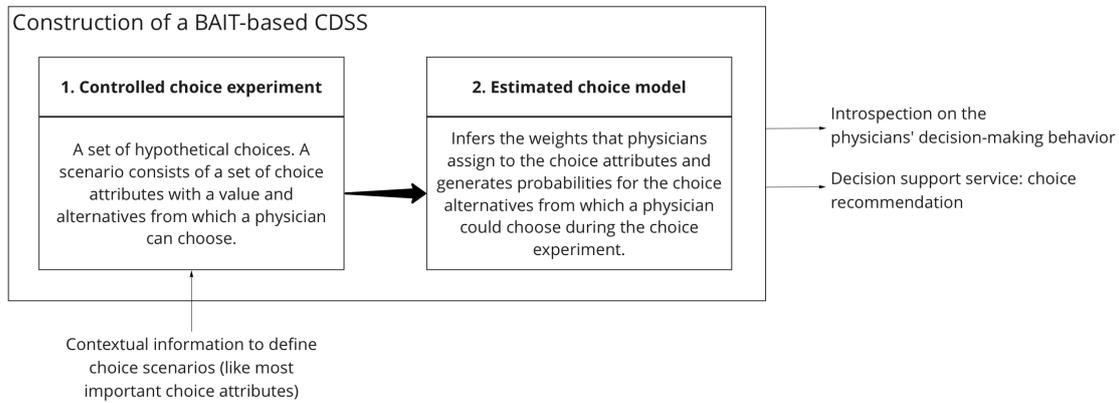


Figure 1.1: The construction of a BAIT-based CDSS.

All hypothetical choice scenarios reflect the situations that the physicians face in their real-life work. Figure 1.2 presents two examples of the hypothetical choice scenarios. As explains section 1.2.1, a set of choice attributes formats the scenarios. For each scenario, the physician chooses between the potential clinical operation alternatives given the characteristics of a patient (Bech, 2003). For instance, a physician chooses to perform a heart surgery or not. The use of hypothetical scenarios brings the CDSS developer in the position to design the choice scenarios (Bech, 2003). By doing so, the developer can control the correlations between attributes and avoid the presence of unobserved attributes that influence respondent's choice (Boyce & Williams, 2015; Helveston, Feit, & Michalek, 2018). As a result, the developer only informs a BAIT-based CDSS about the most relevant attributes in the particular choice task that the CDSS will assist.

Scenario 1: Would you choose for ICU uptake in case of the following patient?		Scenario 2: Would you choose for ICU uptake in case of the following patient?	
Rehousing possibility (regional/national)	No	Rehousing possibility (regional/national)	Yes
ICU capacity	No	ICU capacity	Royal (>2 beds)
Acute condition patient	COVID, not extremely hypoxic, progressive disease.	Acute condition patient	Respiratory threatened: direct incubation required.
Age	70	Age patient	30
Comorbidity: Cognitive impairment	Limited	Comorbidity: Cognitive impairment	Average
Comorbidity: Heart/veins	Limited	Comorbidity: Heart/veins	Average
Comorbidity: Pulmonary	Limited	Comorbidity: Pulmonary	Limited
Comorbidity: Kidney	Limited	Comorbidity: Kidney	Average
Comorbidity: Liver	Limited	Comorbidity: Liver	Average
Comorbidity: Immune system	Limited	Comorbidity: Immune system	Average
Pattern COVID pneumonia	Progressive	Pattern COVID pneumonia	COVID + complications
Body Mass Index	>40	Body Mass Index	20-40
Frailty	5	Frailty	1 and 2
Treatment preference of patient	No uptake	Treatment preference of patient	Uptake

Figure 1.2: Two choice scenarios for a choice experiment on ICU uptake.

The second step in the development of a BAIT-based CDSS is to use the answers of the physicians on the hypothetical scenarios to estimate a choice model (see the right box in Figure 1.1). Based on the experiment choices, a choice model reveals the weights that the group of physicians attaches to the different choice attributes at a specific point in time (de Freitas et al., 2019; J. J. Louviere

et al., 2010). The model thus reflects the decision-making strategy of a group of physicians on a specific choice task. An estimated choice model can predict the future choices of physicians (J. J. Louviere et al., 2010). The prediction that a BAIT-based CDSS gives forms a recommendation on a new choice task defined in terms of the same choice attributes as the hypothetical choice scenarios (ten Broeke, 2020). This means that a BAIT-based CDSS is limited to recurring and structured decisions. As a result of the two development steps, a BAIT-based CDSS offers the following two services (see Figure 1.1):

1. *Introspection on the physician's decision-making behaviour.* The direct result of the formulation of the choice model is an overview of the weights physicians implicitly assign to choice attributes. Analysing the weights estimated for all choice attributes represent the trade-offs physicians make between the choice attributes (Ten Broeke et al., 2021). By doing so, the BAIT-based CDSS generates an improved understanding of physicians' implicit decision-making rules, strategies, and behaviour. The remainder of this report refers the physicians' implicit decision-making rules, strategies, and behaviour as decision-making behaviour. The act of investigating this behavior refers to introspection. Since research shows that it is hard for physicians to explain their logic behind decisions (Wagholikar, Sundararajan, & Deshpande, 2012), introspection already has an added value by generating a starting point for valuable discussions among physicians or an enhanced understanding of how to improve decision-making (Ten Broeke et al., 2021).
2. *Decision support service.* A BAIT-based CDSS allows a physician to enter a real-life choice with regard to a particular patient. This means that a physician shows the choice task regarding a patient to the group of physicians 'behind' the BAIT-based CDSS – the physicians from whom the expertise was used to develop the BAIT-based CDSS. The system then calculates the choice probability. The prediction that a BAIT-based CDSS gives on new choice tasks represents the percentage of physicians that would vote in favour of a treatment. For instance, if a BAIT-based CDSS gives a choice probability of 80%, this means that 80% of the physicians would vote in favour of the alternative. At the same time, this means that 20% of the colleague physicians would not continue with the treatment. If the internal agreement among the pool of colleagues is high, a physician will feel strengthened regarding his or her decision. If the internal agreement is low and the BAIT-based CDSS shows heterogeneous opinions or if the prediction is not in line with the opinion of the physician consulting the recommendation, the physician is signalled and may be triggered to reconsider the decision.

1.3 The dynamic application context of CDSSs

The BAIT-based CDSS is static. Static means that the CDSS can only be developed based on the available knowledge at a specific point in time. However, healthcare decision-making contexts where CDSS are applied are highly dynamic (Bennett & Doub, 2016; Gorzeń-Mitka & Okreglicka, 2014; Miah, Blake, & Kerr, 2020; Prezenski, Brechmann, Wolff, & Russwinkel, 2017; G. Zhang, Xu, & Li, 2012). A decision-making context is defined by the space consisting of all factors that influence the strategy used for a particular choice task. The clinical knowledge driving physicians' decision-making develops, and the contextual conditions under which physicians make decisions continuously change (Castaneda et al., 2015; El-Sappagh & El-Masri, 2014; Miah et al., 2020; Lyman, Cohn, Bloomrosen, & Detmer, 2010). Information central to the decision-making process is thus not static but changes over time (Pérez, Cabrerizo, & Herrera-Viedma, 2010). Physicians' decision-making depends on the outcomes of previously made decisions, on implications in the context that result from these outcomes and on events outside of the physicians' control (Bennett & Doub, 2016; Brehmer, 1992; Edwards, 1962; Hong, Wang, & Lin, 2010). Bennett and Doub (2016) explain that a physician thereby learns progressively and makes decisions while being influenced by, for example, the consequences of previously made decisions or by new regulations and innovative technologies.

Clinical choices involve the incorporation of information that keeps changing (Mahiddin, Othman, & Bakar, 2017). There are three types of dynamics in a healthcare decision-making context.

1. Clinical knowledge develops, and the contextual conditions under which physicians make decisions change every day (Miah et al., 2020). New knowledge about diseases and treatments continuously emerges because the development of clinical sciences is continuous and takes place worldwide (Sanchez et al., 2011; Tomaszewski, 2012). Consequently, the number of

scientific articles testing the effectiveness of treatment and quality of healthcare increases (Tomaszewski, 2012). As Sanchez et al. (2011) give as an example, a treatment could be rendered irrelevant by a new discovery that supersedes it.

2. Accepted norms and regulations may change the possibilities of physicians by restricting specific treatments.
3. The pool of physicians making the decisions with regard to a specific treatment may change.

As a result of these dynamics, the information central to the decision-making process is not static but changes over time (Pérez et al., 2010). Accordingly, the decision-making strategy of physicians and the resulting choices evolve (Zhu et al., 2020). If CDSS developers do not process these changing contextual conditions in the decision support, recommendations become inaccurate and do not reflect intended goals (Lenz, 2020). CDSS's are primarily dependent on the clinical information based on which they are modelled and can thus only be as effective as the quality and relevance of this information (Khalifa, 2014; Osop & Sahama, 2019). When a CDSS depends on unchanged information over time, the performance of a CDSS is known to degrade over time because its application context does not remain in a stationary state (Klinkenberg & Rüping, 2002). Consequently, the CDSS is inactive and will become unserviceable over time (El-Sappagh & El-Masri, 2014).

Therefore, (Bennett & Doub, 2016) argue that CDSSs should mimic the way physicians evolve within decision-making over time. Arnott (2006, p.2) agrees and states that a CDSS must be updated frequently during ongoing use to “track changes in the problem, user, and decision-making environment because these factors are inherently volatile”. Similarly, Vahidov and Kersten (2004) argue that all future CDSSs should be capable of adapting to changes in their environment. Accurate and up-to-date information can optimize the decision-making process because the new knowledge in healthcare inspires the physicians’ decisions (Gago & Santos, 2008; Kruse, Goswamy, Raval, & Marawi, 2016). Therefore, the successful adoption of a BAIT-based CDSS requires transforming the static structure into a version that remains accurate over time. When a CDSS remains accurate over time, it is applicable in dynamic healthcare contexts.

1.4 Research objective

Based on the introduction about BAIT-based CDSSs and the dynamic healthcare contexts, this section presents the objective of this research on transforming the static BAIT-based CDSS into a version that remains accurate over time. First, subsection 1.4.1 describes the value of BAIT-based CDSSs in more detail. Next, subsection 1.4.2 defines the concept of architecture and points out the value of architecture. Based on these two sections, subsection 1.4.3 explicates the goal of this research.

1.4.1 The value of a BAIT-based CDSS

Previous research shows that the characteristics of a BAIT-based CDSS are promising for the support of healthcare decision-making (Ten Broeke et al., 2021). The main reason is that the BAIT approach is transparent. A transparent approach implies that a BAIT-based CDSS is understandable, interpretable, tractable, and revisable (Sutton et al., 2020). A transparent CDSS is of great value in healthcare contexts because decisions need to be explainable (Sutton et al., 2020). Physicians are hesitant to use a CDSS when the reasoning behind a recommendation of this CDSS is opaque (Holst et al., 2000; Wang et al., 2021). Physicians better accept transparent support systems because these systems allow them to understand the recommendations of a CDSS and minimize the risk of misinterpretation (Sinha & Swearingen, 2002; Holzinger, Biemann, Pattichis, & Kell, 2017; Curcin, Fairweather, Danger, & Corrigan, 2017; Sutton et al., 2020). Raghupathi and Raghupathi (2014) even states that systems must be transparent and simple to succeed in healthcare. Therefore, Korva, Porter, O’Connor, Shaw, and Brinke (2013) argues that there is a need for CDSSs with transparent algorithms.

The existing CDSS solutions introduced in section 1.1, have limited potential for application in dynamic healthcare contexts. NKBS’s can incorporate subtle adjustments over time but operate as “black-boxes” meaning their reasoning is not transparent (Rudin, 2019). In healthcare, the potential existence of unobservable biases in algorithms leads to ethical concerns (Xafis et al., 2019). KBS’s are transparent but are captured in a predefined, fixed mold of rules (Bennett & Doub, 2016). Due to the fixed and stifle if-then-else rules, KBS’s cannot incorporate subtle

adjustments over time. To maintain KBS's, knowledge and software engineers need to map and implement decision-making changes. This maintenance need leads to time-consuming system development (Yan, 2018)). Because it is hard to maintain KBS's, they are inadequate for application in contexts involving many interacting and changing variables.

The BAIT approach allows a CDSS to slightly tune its underlying model while retaining its transparency. The BAIT approach enables this transparent tuning because BAIT is grounded in DCM: a choice model can let its model parameters vary according to the changing preferences of physicians over time (Danaf, Becker, Song, Atasoy, & Ben-Akiva, 2019; Lachaab, Ansari, Jedidi, & Trabelsi, 2006; Siddarth, Bucklin, & Morrison, 1995). As a result, a BAIT-based CDSS has the potential to keep up-to-date with the changes in the decision-making context and thereby provide accurate recommendations over time. Because of the importance of transparent decision support in healthcare decision-making contexts, the BAIT approach forms an important innovation for CDSS developers who aim to satisfy the decision support needs of physicians.

1.4.2 The value of architecture design

For CDSS developers to develop transparent CDSSs for application in dynamic healthcare contexts, developers should be able to design CDSSs based on the BAIT approach. Many CDSS development projects start with creating a system architecture. A CDSS architecture is a formal definition of all the information system and technology components relevant to developing a CDSS (Power, 2002). A CDSS architecture includes the components used to manage information and communication and the overall configuration that integrates the various components (Applegate, McFarlan, & McKenney, 1996). For this research, a CDSS architecture is a formal description of all CDSS components that guides developers in implementing a CDSS. There are six principal reasons for the popularity of designing a system architecture in the system design process (Garlan, 2000).

1. *Understandability.* A CDSS architecture explains the CDSS's structure at a level of abstraction at which a developer can easily comprehend the architecture (Perry & Wolf, 1992; Garlan & Shaw, 1993). At the same time, it shows the rationale for specific design decisions, and the restrictions on the CDSS (Garlan, 2000). As a result, a CDSS architecture supports the collaboration among developers and stakeholders, it allows designers to argue the extent to which the CDSS satisfies requirements, it improves the planning of the implementation, and it stimulates the ability to communicate the (future) key aspects of the CDSS to developers and end users of the CDSS. By doing so, it provides a tractable guide to the overall CDSS (Garlan, 2000; Power, 2002).
2. *Reuse.* Other design projects can reuse a CDSS architecture as a blueprint that suggests the construction and composition of the CDSS (Garlan, 2000). The reuse can also include parts of the architecture.
3. *Construction.* A CDSS architecture forms a blueprint for developing a CDSS by showing what components need to be in place and how they are related. In addition, an architecture can help ensure a complex CDSS satisfies the main requirements regarding the relationships between many components (Garlan, 2000). Besides, from a technical point of view, CDSS architecture allows evaluating technology options on how they will work and make proper design decisions accordingly (Power, 2002).
4. *Evolution.* A CDSS architecture exposes the dimensions along which the CDSS may evolve over time. This exposure allows for a better understanding of the external changes that may impact the CDSS's application and the related costs of required adjustments.
5. *Analysis.* An architecture allows to analyse aspects of the CDSS as a matter of evaluation (Buchgeher & Weinreich, 2014). For instance, by focusing on the CDSS's consistency, the CDSS's conformance to requirements and restrictions, a dependency analysis, and the CDSS's applicability in a specific domain (Abowd, Allen, & Garlan, 1993; Allen & Garlan, 1994; Clements, Bass, Kazman, & Abowd, 1995; Magee, Dulay, Eisenbach, & Kramer, 1995; Stafford, Richardson, & Wolf, 1998).
6. *Management.* A CDSS architecture allows for an early evaluation of the CDSS. An evaluation may lead to a clearer understanding of requirements, suitable strategies for the CDSS implementation, and risks of the CDSS implementation (Boehm, Bose, Horowitz, & Lee, 1995). Modifying an architecture based on evaluation outcomes is relatively easy, while revising an implemented CDSS may be costly.

An architecture design is especially of value in the context of a BAIT-based CDSS for two primary reasons. First, BAIT is a novel technology. Transforming the static BAIT-based CDSS into a dynamic version therefore goes hand in hand with new design challenges. An architecture design that allows for evolution and analysis better suits the phase of the BAIT approach. Second, the healthcare contexts in which BAIT-based CDSSs are to be applied are diverse (Eapen, 2021). Consequently, preferences of physicians regarding the CDSS will differ per context. Moreover, because healthcare contexts are dynamic, these preferences may also vary over time. Therefore, there is no one-size-fits-all solution for the transformation of a BAIT-based CDSS into a version that retains its accuracy over time. Instead of a prefabricated CDSS design, CDSS developers benefit from an architecture that maps the relevant CDSS components and guides developers in creating a context-specific CDSS that satisfies the needs in the particular decision-making context.

1.4.3 Filling the gap of guiding design principles

The entities possessing the knowledge and skill to create architectures for CDSS developers are referred to as architecture designers. Although architecture designers possess the required design knowledge and skill, they will encounter a challenging process when designing an architecture of a BAIT-based CDSS. This expectation is rooted in the following four reasons:

1. The BAIT approach forms a novel technological innovation for which technical and social feasibility in various dynamic healthcare decision-making contexts is still to be proved. In addition, the potential implications of such an unexplored technology are yet unclear and may give rise to unexpected events.
2. The architecture should be sufficient for the development of CDSSs for varying healthcare decision-making contexts (Eapen, 2021). The physicians' preferences in the variety of application contexts are all to be addressed by the architecture.
3. The architecture design relates to knowledge on both CDSS architecture design and the econometric field of discrete choice Modeling (DCM). No research has combined these fields before. Therefore, there is no fundamental material in place providing examples based on both areas.
4. The architecture design is dependent on needs at two levels: the developer who will use the architecture to create a CDSS and the physicians who will use the CDSS that the architecture defines. As such, the architecture designer should perform research and design activities while incorporating considerations at both levels.

Architecture designers need support in tackling the challenges mentioned above to develop transparent and accurate CDSS's for healthcare contexts (see subsection 1.4.1). Therefore, groundwork on designing a dynamic BAIT-based CDSS architecture is necessary. This groundwork should provide rules that a designer can follow to arrive at the intended architecture (Winter & Aier, 2011). In architecture design, these rules are known as architectural design principles (Lindstrom, 2006). "Design principles are created to codify and formalize design knowledge so that innovative, archival practices may be communicated and used to advance design science and solve future design problems, especially the pinnacle, wicked, and grand-challenge problems that face the world and cross-cutting markets" (Fu, Yang, & Wood, 2015, p. 1). Design principles define the guidelines for developing architectures that aim to increase the chance of reaching a successful artefact (Fu et al., 2015). They are thus a form of knowledge explication. Over the years, design principles have even become the predominant manner to capture abstract knowledge about the development of similar information systems (Kruse et al., 2016). As such, they are "normative, reusable and directive guidelines, formulated towards taking action by the information system architects" (Bharosa & Janssen, 2015, p. 472). On its own, an architecture is rather descriptive, essentially passive, and cannot provide operational design guidance (Hoogervorst, 2009). Stimulating designers to create accurate BAIT-based CDSS architectures requires an explication of the principles underlying this architecture.

However, there are no design principles that can function as groundwork for dynamic BAIT-based CDSS architecture designers. Because the value of transparent decision-making support in healthcare contexts is tremendous and has a growing societal relevance, this research aims to formulate design principles that guide the design of dynamic BAIT-based CDSS architectures. Hence, the goal of this research is to:

“develop and test design principles for an architecture of a dynamic BAIT-based CDSS”.

In the remainder of this report, the concept “dynamic CDSS” points to a CDSS that incorporates newly available choice information in the decision-making context and, thus, improves the quality of its recommendations during ongoing use. As a BAIT-based CDSS consumes choices as input data, the concept “dynamic BAIT-based CDSS” refers to a version of the BAIT-based CDSS that incorporates new choice information. These choices reflect the changes in a healthcare decision-making context. To conclude, this research focuses on developing design principles for designing an architecture that enables a BAIT-based CDSS to integrate new choices during ongoing use. This chapter introduced multiple stakeholders who benefit from or can inform these design principles. To give an overview of the stakeholders, Figure 1.3 maps these stakeholders.

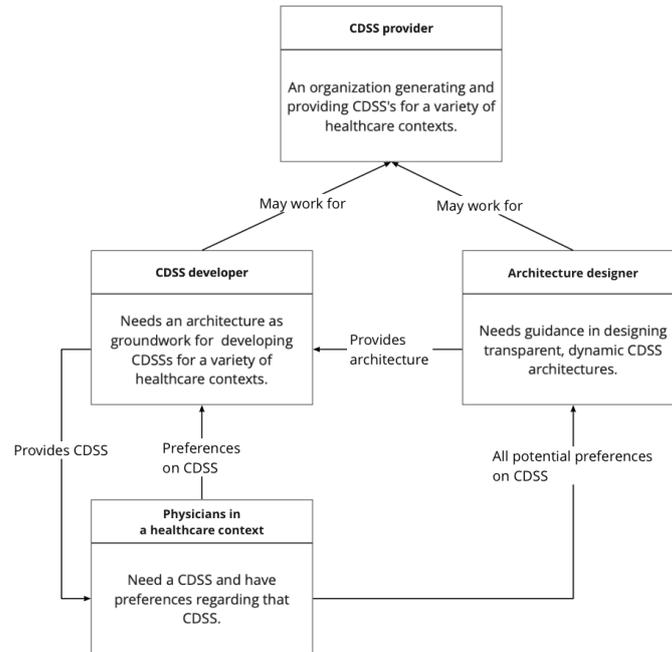


Figure 1.3: Overview of the stakeholders.

1.4.4 Research questions

Achieving the research objective requires the formulation of research questions. Answering these questions results in the knowledge necessary in reaching the purpose of a research (Verschuren, Doorewaard, & Mellion, 2010). Based on the research objective (see section 1.4), the overall question of the research is framed as follows:

To what design principles should a system architecture of a dynamic BAIT-based CDSS adhere?

Design Principles are typically derived inductively from experiences, examples, or empirical evidence (Bell, Hoadley, & Linn, 2004; Fu et al., 2015). As such, the knowledge gained during an architecture design process can inspire the formulation of architecture design principles (Fu et al., 2015). Therefore, answering the main research question requires obtaining knowledge while designing a dynamic BAIT-based CDSS architecture. This design and knowledge acquisition process is structured following six sub-questions. The continuation of this paragraph presents these sub-questions.

Existing CDSS architectures used to design CDSSs also deal with contextual changes and can, therefore, function as an initial example. Hence, the architecture design starts with investigating the components worth considering when designing a dynamic CDSS. This investigation results in the definition of a framework that specifies the concepts that are at stake when designing a dynamic CDSS architecture. The framework helps to set the focus areas of this research. To this end, the first step is answering the following question:

1. What are the main components of a dynamic CDSS architecture?

The technological foundation of a BAIT-based CDSS is significantly different from that of other CDSSs. Therefore, the inheritance of the main components of existing CDSS architectures to a BAIT-based CDSS architecture may be restricted. It is relevant to find out if and how the DCM foundation of a BAIT-based CDSS shapes the design of the main components identified. Thus, the subsequent research step is to extend the framework by formulating how the theories, practices, and characteristics of DCM shape the main components identified. The need for this step leads to the following sub-question:

2. How is the design of the main components of a dynamic CDSS architecture shaped by the theories, practices and characteristics of DCM?

Having in place the critical components of a dynamic CDSS and the restrictions posed to these components by DCM, a further outline of the design is realized by selecting architectural requirements. Requirements can function as detailed descriptions of what an architecture user ultimately wants from the architecture design (Dym, Little, Orwin, & Spjut, 2004). Accordingly, the requirements form input for the architecture design as is desired by potential users. As such, the requirements function as a hypothesis regarding the sought for design principles. The need for requirements leads to the following sub-question:

3. What are the requirements for a system architecture of a dynamic BAIT-based CDSS?

The formulated requirements function as guidelines during the design process of the architecture. However, an incorrect formulation of the requirements results in delays or even mistakes in the architecture design (Shah & Patel, 2016). Designing an architecture in line with the requirements allows assessing whether the requirements successfully lead to an architecture of a dynamic BAIT-based CDSS. To this end, the following sub-question is relevant:

4. What does a system architecture of a dynamic BAIT-based CDSS look like?

With an architecture solution in place, it is not yet ascertained that the requirements together effectively guide the design of the desired architecture. Therefore, the next step is to assess the usefulness of the architecture. The need for this assessment leads to the following research question:

5. To what extent does the designed architecture help to develop an effective tool for guiding the development for a dynamic BAIT-based CDSS?

In the process of answering the sub-questions mentioned above, lessons on designing the architecture will arise. These lessons are of interest to future designers facing similar design challenges and shed light upon seemingly incongruent perspectives. Therefore, the lessons are relevant to design science research and designers facing a design problem belonging to a similar class of problems. The formulation of design lessons allows making a first move towards formalizing knowledge for the broader class of problems. The final sub-question thereby is:

6. Considering the requirements and the evaluation, what are the lessons about how to design architectures of a dynamic BAIT-based CDSS?

Combining the answers to all of the sub-questions generates a solution to the main research question. Section 1.5 presents the research design that structures the process of answering these sub-questions and the main question. This section also informs on the information and data collection methods.

1.5 Research approach

This section presents the approach that guides the research. First, subsection 1.5.1 introduces the approach used to structure the study. Moreover, it underpins the selection and highlights the drawbacks of the approach. Second, subsection 1.5.2 describes the main research phases to shed light upon the research flow.

1.5.1 Selection of the research approach

Two aspects are important in the explanation of the design approach selection. First, section 1.5.1 explains the choice for a particular design strategy. Second, subsection 1.5.2 shows how this strategy inspired the choice for a research framework.

Selection of the design strategy

As explained in subsection 1.4.4, the process of designing a dynamic BAIT-based CDSS architecture informs an answer to the main question. As such, this research is design-oriented. Design as a science refers to knowledge “in the form of constructs, techniques, methods, models and well-developed theory for performing the mapping between the know-how for creating artefacts and satisfying given sets of functional requirements” (V. K. Vaishnavi & Kuechler, 2015, p. 3). The research that is fundamental to developing this kind of knowledge is Design Science Research (DSR) (Lavasani, Hossan, Asgari, & Jin, 2017). DSR is a valid and relevant approach in Information Systems, the area of CDSSs (Arnott, 2006; A. Hevner & Chatterjee, 2010).

A DSR project can follow two strategies (Iivari, 2015). The first strategy begins with a general solution concept that enables the instantiation in multiple, specific healthcare contexts. The second strategy drives the researcher to solve a problem of one particular client with a concrete artefact. During the development, the researcher is active in the context of this client. By doing so, the researcher can collect prescriptive knowledge informing a general solution concept that addresses a class of problems (Iivari, 2015).

The design challenge defined in section 1.4 stems from a practical problem experienced by the start-up Councilyl. Councilyl has been developing and improving the BAIT approach since April 2020. Because BAIT is a novel technology, Councilyl has been the only organization to apply BAIT. As such, Councilyl possesses critical knowledge about a BAIT-based CDSS. For that reason, this study will focus on designing an architecture in the context of the start-up Councilyl. By doing so, this study follows strategy 2 (Iivari, 2015): the study starts from solving a problem in a situated context (Councilyl) to then distill general knowledge relevant for answering the main research question.

Selection of the design framework

DSR literature suggests several research frameworks that inform researchers on the necessary design research activities. Sein, Henfridsson, Purao, Rossi, and Lindgren (2011) recently published a new framework that endorses strategy 2 of (Iivari, 2015). The starting point of their work was their intention to combine building, intervention, and evaluation of an IT artefact in a concerted research effort (Keijzer-Broers & de Reuver, 2016). Sein et al. (2011) argue that a design research process does not follow distinctive phases, like Peffers, Tuunanen, Rothenberger, and Chatterjee (2007) and Kuechler and Vaishnavi (2008) suggest, or structurally ordered research steps as A. R. Hevner (2007) proposes. Instead, “the research steps are 1) less structured than that, 2) executed concurrently and 3) can be regarded as an iterative process” (Keijzer-Broers, 2016, p. 40). As a result, they proposed Action Design Research (ADR) as a new design science research method.

ADR is “a research method for generating prescriptive design knowledge through building and evaluating ensemble IT artefacts in an organizational setting” (Sein et al., 2011, p. 4). ADR forms a variant of DSR frameworks that start from a theoretical design problem (A. R. Hevner, March, Park, & Ram, 2004; March & Smith, 1995; V. Vaishnavi & Kuechler, 2004). ADR starts from a specific problem context. Since the core consideration of ADR is how practical problems in a specific setting drive the design of an artefact (Keijzer-Broers, 2016), it seamlessly fits the proposed design project. Two arguments will further substantiate this.

First, the ADR method is helpful in a situation where specific outlines of the solution are yet unclear, and the understanding of the organization will increase along the design process. ADR is beneficial in this situation because it allows the researcher to continuously evaluate a design and improve it while better understanding the organizational needs. Unlike other DSR methods (see (Peffers et al., 2007)), ADR emphasizes that the artefact is to be shaped by both the researcher and organizational actors during the entire design process (Haj-Bolouri, Purao, Rossi, & Bernhardsson, 2018). The organizational intervention becomes a priority instead of a secondary activity as in other DSR methods (Cole, Purao, Rossi, & Sein, 2005). The system architecture design subject to this research is closely bound to the organization Councilyl. As such, it is crucial to stress the role of Councilyl in shaping the design. Because the ADR method drives the researcher to involve the organization, this method forms an adequate basis for this design research.

Second, performing research in an organizational setting may cause the researcher to concentrate too much on developing a solution. This focus on the artefact draws away the researcher’s attention from the production of design knowledge (Maccani, Donnellan, & Helfert, 2014). Because design knowledge can inform a generic solution concept, the production of this knowledge is the ultimate goal of design science research (A. R. Hevner et al., 2004; Kuechler & Vaishnavi, 2008). Therefore, ADR suggests specific tasks on generalization and encourages the researcher to balance between addressing the problem encountered in a particular setting and generating generic knowledge for a broader class of problems (Petersson & Lundberg, 2016)). By doing so, “ADR combines theory with researcher intervention to solve immediate organizational problems. Thus, ADR aims to link theory with practice, and thinking with doing” (Sein et al., 2011, p. 39).

1.5.2 Research flow

This section clarifies how the ADR framework guides the identification and formulation of the design principles. The ADR framework proposes four phases (Problem formulation - Building, intervention and evaluation (BIE) - Reflection and learning - Formalization of learning). All phases draw upon principles. These principles capture the assumptions, values, and beliefs that are important when performing a ADR study. In total, an ADR researcher should adhere to seven design principles. Figure 1.4 gives a visualization of the four ADR research phases and the principles fundamental to each phase. Figure 1.5 shows how the ADR framework drives the organization of the research activities central to this research.

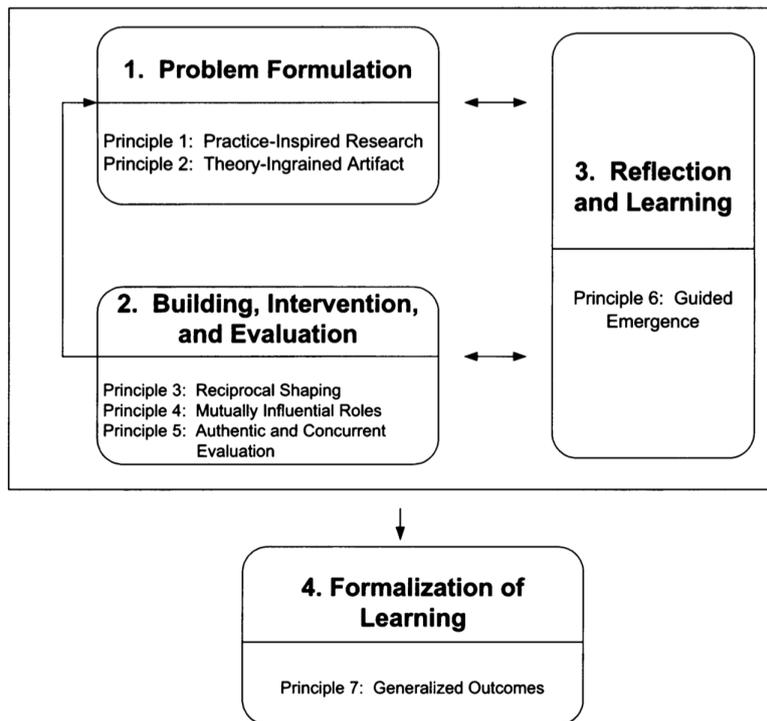


Figure 1.4: The phases of the Action Design Research (ADR) framework.

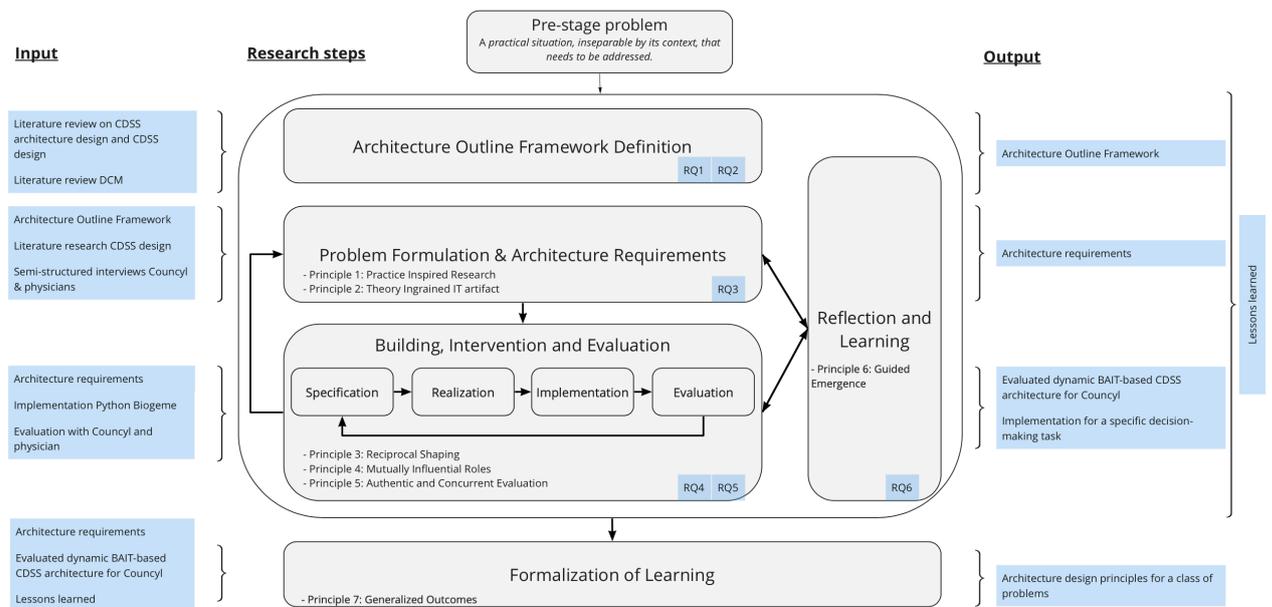


Figure 1.5: Research flow with the input, phases, output, Action Design Research (ADR) principles, and sub-questions.

Phase 0: Architecture design space definition

Although not particularly specified by the ADR framework, the research starts with a literature review on existing architectures for CDSSs. With this step, the research adheres to ADR Principle 2: Theory ingrained artefact (Sein et al., 2011). The CDSS architecture literature gives insight into the principal components of an architecture and shows how researchers tackled similar design problems in the past. The literature findings will be structured in a framework that delimits the design space for creating a dynamic BAIT-based CDSS architecture.

Phase 1: Problem formulation and Architecture Requirements identification

The Problem Formulation phase is about formulating the problem as perceived by the researcher (Keijzer-Broers, 2016). A thorough understanding of the problem enables the designer to attune the solution to the primary user needs (Islam, Weir, & Del Fiore, 2014; Hutton & Klein, 1999). In line with ADR Principle 1, a problem that occurred in practice inspired this research (Sein et al., 2011). The problem description that Council provided is clear enough to function as the starting point of the ADR process. Therefore, the goal of the Problem Formulation phase is not to simply understand what Council's problem is. Instead, the goal is to investigate what Council wants the architecture to be and do in terms of architecture requirements. This knowledge denotes when the architecture design solves the problem experienced by Council.

Interviews are a standard method to gain insight into an initial list of requirements (Johannesson & Perjons, 2014). Interviews allow to engage in dialogue and identify the requirements interactively and creatively (Johannesson & Perjons, 2014) (Agarwal & Tanniru, 1990; Johannesson & Perjons, 2014). Moreover, the direct approach to the elicitation of requirements by asking stakeholders about preferred features gives insight in many requirements in a short time (Johannesson & Perjons, 2014). Hence, interviews form an efficient method. An additional advantage is that interviews will increase the understanding of the interviewed stakeholders about and improve their attitude towards the architecture (Johannesson & Perjons, 2014).

The interview respondents are the key stakeholders of the architecture design. The incorporation of the stakeholders supports ADR Principle 4, Mutually influential roles. This principle emphasizes mutual learning among the different research, and organizational stakeholders (Sein et al., 2011). The key stakeholders are the architecture users – the representatives of Council – and the end user of the CDSS that the architecture describes – physicians. Chapter 5 gives additional details on the organization of the interviews and the respondents.

Interviewing physicians gives rise to two implications. First, the information that physicians provide is about the CDSS they will use. Therefore, the output of the interviews with the physicians needs to be translated into requirements at the architecture level. Second, identifying the needs

of end users regarding new technological solutions is a complex process (Maiden & Hare, 1998). Asking end users about their needs is not as straightforward as expected (Hyysalo & Lehenkari, 2003; Pitts & Browne, 2007). End users often do not understand the technical terms and find it hard to make explicit what they really need (Keijzer-Broers & de Reuver, 2016). To deal with these two implications, phase 1 involves an additional data collection method: the interpretation and translation of the interviews were guided and strengthened with suggestions found in the architecture design literature. Subsection 3.2.2 explains the combination of interviews and literature in more detail.

The first phase gives insight into the initial list of requirements. A researcher will gain new insights regarding the design along an ADR process (Sein et al., 2011). Therefore, the researcher needs to refine the initial set of requirements along the design process.

Phase 2: Building, Intervention Evaluation

The goal of the second phase is to use the findings of the previous phases to design the architecture (Keijzer-Broers & de Reuver, 2016). The ADR method states that this design process consists of building and constantly evaluating the design in repeated cycles. The design process consists of three cycles. Each cycle takes three weeks. Developing the architecture in a sequence of cycles is in line with principle 3, Reciprocal shaping (Sein et al., 2011).

A drawback of the ADR method is that it does structure the second phase well. Therefore, the research follows the design steps proposed by the framework of Verschuren and Hartog (2005). Figure 1.6 presents this framework. The framework of Verschuren and Hartog (2005) matches the highly iterative character of the ADR method. Verschuren and Hartog (2005) argue that the evaluation of the design should happen continuously. An evaluation may indicate that the artefact does not yet fulfil the organizational requirements and that another iteration of the cycle will be needed. The “Prototype” step is referred to as “Realization”, a term Verschuren and Hartog (2005) use for the explanation of this phase. This replacement avoids confusion with the implementation phase. Accordingly, the BIE phase follows the specification, realization, implementation, and evaluation steps. Because the requirements were needed to structure the architecture design in phase 2, this research covers the requirement identification step (step 2 in the framework of Verschuren and Hartog (2005)) already in phase 1 (section 1.5.2). Therefore, phase 2 does not copy the requirements step. The continuation of this section describes the research steps part of each design cycle.

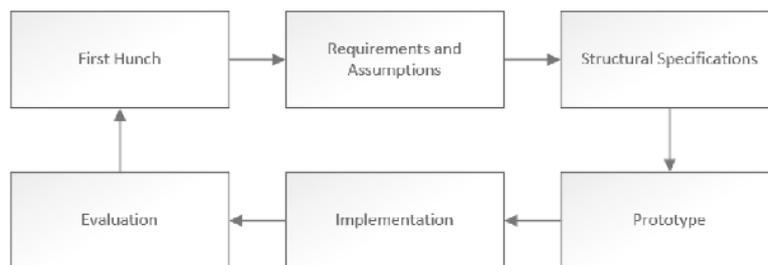


Figure 1.6: Design Cycle of Verschuren and Hartog (2005).

Each cycle starts with the specification. During the specification, the requirements are translated into specifications that can be realized in the architecture design (Verschuren & Hartog, 2005). The realization of some requirements into an architecture gave rise to design challenges. A design challenge refers to the situation in which additional research is necessary or a trade-off exists between requirements. In case of a trade-off, the realization of one requirement hindered the realization of another requirement. Further investigation was either by conducting sessions with Council or by consulting literature providing relevant theoretical concepts and premises. The Realization step focuses on forming an architecture design by combining the formulated specifications and the realization of the remaining requirements.

Each design cycle concludes with an implementation and an evaluation of the architecture design. The implementation covers the development of components that are only observable during run time. To this end, the implementation consists of a set of programmed and visualized CDSS

functionalities. The evaluation is twofold. A static-oriented evaluation assesses the usefulness and usability of the architecture for Council. A dynamic-oriented evaluation assesses whether the architecture describes the CDSS functionalities as desired by Council. This part of the evaluation uses the implementation outcomes. Each BIE cycle, the architecture implementation and evaluation generate new insights regarding the situated solution. These insights further shape the architecture design. As such, the output of the implementation and evaluation functions as input for the new design cycle.

Phase 3: Reflection and learning

The goal of the third phase is to make the first step from building a solution for Council to applying the situated learning to a broader class of problems (Keijzer-Broers, 2016). Therefore, this phase focuses on identifying lessons learned throughout the research process. The relevance of the lessons is that they can support future designers who face similar design challenges.

Four categories of lessons exist: lessons on the problem formulation, emerging artefact, theories ingrained in the artefact, and the design process (Sein et al., 2011). All lessons stem from unforeseen challenges encountered throughout the design research in the situated context. Since these challenges occur throughout the entire design process, this phase happens in parallel with the phases mentioned above. As such, it forms a continuous process. The design decisions and all implications for the design process were written down in a logbook to support this continuous reflection. The research process ends with creating a list of the key lessons. By doing so, the research adheres to the sixth principle of Guided emergence, which emphasizes the importance of incorporating the outcome of addressing the previous five principles in the design.

Phase 4: Formalization of learning

The goal of the final phase is to further develop the situated learning into generic concepts that solve a broader class of problems. Here, it is about the formalization of the research findings. To realize the goal of the final phase, the architecture requirements found in the situated context are generalized into design principles that are applicable for a broader class of problems. To be able to make this move, three levels of generalization need to be considered (Sein et al., 2011).

1. A generalization of the problem instance
2. A generalization of the solution concept
3. Derive design principles from the design research outcomes

Accordingly, the formalization starts with articulating the problem of designing a dynamic BAIT-based architecture as a class of problems in CDSS architecture design. The next step is to frame the designed architecture as a representation of a class of solutions. Finally, phase 4 focuses on formulating reusable design principles based on the identified architecture requirements. The lessons learned guide this formulation (see section 1.5.2). By doing so, the final set of design principles is not only inspired by requirements but also by the lessons that were learned during the design process. As a result, the design principles have the power to prevent future designers from encountering similar challenges and making similar mistakes. Therefore, the resulting design principles function as recommendations for designing comparable CDSS architectures in the future. This phase draws on principle 7, Generalized outcomes. This principle emphasizes that generalization is challenging but essential since the artefact has been developed for a specific context (Sein et al., 2011).

1.6 Summary chapter 1

The goal of chapter 1 is to give an introduction to the research presented in this report. Physicians make complex decisions on clinical treatments that directly influence patients' well-being daily. To deal with such complex decisions and to minimize decision errors, physicians show a growing interest in Clinical Decision Support Systems (CDSS). BAIT forms a new approach to decision support. The introduction of the BAIT-based CDSS is promising because it allows for the design of transparent CDSSs. Physicians greatly value the transparency of CDSSs.

However, the development of BAIT-based CDSSs has not progressed further than a static version. As a result, the recommendations of a BAIT-based CDSS are not informed by new knowledge available in the decision-making context. This conflicts with the dynamic nature of healthcare decision-making contexts. Therefore, the successful adoption of a BAIT-based CDSS

requires transforming the static BAIT-based CDSS into a version that provides accurate choice recommendations over time. This research report uses the term dynamic BAIT-based CDSS to refer to a BAIT-based CDSS that can update with new knowledge and retains its accuracy over time.

The availability of dynamic BAIT-based CDSS architectures could foster the development of transparent and dynamic CDSSs. However, no principles guiding the design of such architectures exist. Therefore, this research aims at developing and testing design principles for an architecture of a dynamic BAIT-based CDSS. To realize this goal, an answer to the following research question is necessary: What design principles should a system architecture of a dynamic BAIT-based CDSS meet?

Design principles can be derived inductively from design experiences, examples, or empirical evidence. To work towards an answer to the research question, this research follows the Action Design Research (ADR) framework (Sein et al., 2011). The ADR framework allows to identify and test the requirements of a dynamic BAIT-based CDSS architecture in a situated action context. The situated context is the start-up Councilyl. Councilyl currently produces BAIT-based CDSSs and thus possesses the critical knowledge for the architecture design process. The ADR framework proposes four design phases: the problem formulation and Architecture Requirements identification, BIE (Building, Intervention Evaluation), reflection and learning, and formalization of learning. The last phase generalizes the architecture requirements identified and tested in the situated context. By doing so, a set of design principles is found that address a broader class of problems.

Chapter 2

Definition of the architecture design space

Existing architectures used to develop dynamic CDSSs already incorporate techniques to update the CDSS according to medical developments. Therefore, studies in the field of CDSS design and CDSS architecture design can function as an initial example. However, the technological foundation of a BAIT-based CDSS that applies Discrete Choice Modeling (DCM) is significantly different from that of traditional CDSSs. Therefore, this chapter aims to identify the main components of a dynamic CDSS and investigate how DCM theories, practices, and characteristics shape these components. To this end, this chapter works towards an architecture design space framework and presents the theories and practices that are relevant when designing a dynamic BAIT-based CDSS architecture. First, section 2.1 distinguishes the main components that CDSS design and CDSS architecture design literature suggest. Given the purpose of this research, the focus of the framework is on the CDSS components necessary to update a CDSS so that the CDSS bases its recommendations on recent clinical information. Second, section 2.2 describes how DCM theories, practices, and characteristics shape the design of these main components. The chapter concludes with the architecture design space framework that summarizes the main design considerations in the context of a BAIT-based CDSS architecture in section 2.3. By outlining the design space of the main components in the context of a dynamic BAIT-based CDSS architecture, this chapter provides an answer to the first and second sub-question:

1. **What are the main components of a dynamic CDSS architecture?**
2. **How is the design of the main components of a dynamic CDSS architecture shaped by the theories, practices and characteristics of DCM?**

2.1 Architecture components of a CDSS

This section introduces the main components of a dynamic CDSS. The architecture of a CDSS essentially consists of two key parts: intelligent internal software that runs all the processes necessary for qualitative decision support and systematic Human-Computer Interaction (HCI) (Power, 2002; Yun, Ma, & Yang, 2021). The design of the HCI component depends on the intelligent internal software of the CDSS: the HCI component covers the interaction between a physician and a CDSS to enable physicians to achieve their goals with the functionalities that the underlying intelligent internal software provides. The addition of new CDSS functionalities that involve human interaction goes hand in hand with the addition of HCI components. Therefore, the design of a dynamic BAIT-based CDSS architecture design does not only concern components that make it technically possible to update a CDSS. The design also concerns establishing the interaction between a physician and a BAIT-based CDSS. Section 2.1.1 introduces the essential intelligent internal software components of dynamic CDSS. Section 2.1.2 defines the aspects of the HCI design for a dynamic CDSS. Finally, Figure 2.3 gives an overview of all primary components to consider when designing an architecture of a dynamic CDSS.

2.1.1 Intelligent internal software

Literature on CDSS architecture design acknowledges that for a CDSS to retain its accuracy and relevancy for physicians over, updating the CDSS is inevitable. To prevent a CDSS from becoming unserviceable over time, the information fundamental to a CDSS must be up-to-date. Therefore, a CDSS designer needs to refresh this information regularly (El-Sappagh & El-Masri, 2014; Lyman et al., 2010; Osop & Sahama, 2019). Because a significant quantity of clinical research is published on an ongoing basis (Gluud & Nikolova, 2007; Kruse et al., 2016), continuous efforts are essential to sustain the accuracy of this information (Klein Koerkamp, 2019; Trivedi et al., 2009). To account for these constant maintenance efforts, existing CDSS architecture literature proposes three internal software components: an adaptive knowledge base so that the CDSS accepts new information, a model update engine so that the CDSS continually updates according to that new information, and a model quality monitor so that the CDSS timely detects any decrease in performance in a changing context (Chen et al., 2007; El-Sappagh & El-Masri, 2014; Gago & Santos, 2008; Power, 2002; Velickovski et al., 2014; Zikos & DeLellis, 2018). The sections below give a description of each component.

Adaptive knowledge base

CDSS architectures commonly include a knowledge base (Power, 2002; Tariq & Rafi, 2012; Velickovski et al., 2014). This base contains a collection of data organized for easy access and analysis by the CDSS (Power, 2002). Research on evidence-adaptive CDSSs focuses on the adaptivity of these knowledge bases. The concept of "evidence" refers to the literature- or practice-based information that the CDSS uses to infer the relevant decision-making patterns (Khalifa, 2014). A CDSS is evidence-adaptive if the CDSS has the mechanisms to incorporate novel information into the knowledge base and the knowledge base thus always reflects the latest information in the decision-making context (Afzal et al., 2014; Sim et al., 2001). Without updating the knowledge base over time, the CDSS generates recommendations using inaccurate information. As a result, the failure rate of the CDSS will increase over time (Gago & Santos, 2008).

Model update engine

A CDSS generates recommendations with a model that is built based on the information in the knowledge base (Power, 2002; Tariq & Rafi, 2012; Yao & Kumar, 2013). In the remainder of this report, this model is referred to as the choice recommendation generator model. The above-mentioned evidence-adaptive CDSS's are only dynamic if the novel information in the adaptive knowledge base is also processed during an update of the choice recommendation generator model (Osop & Sahama, 2019). Therefore, a dynamic CDSS needs a component that updates the recommendation generator model with new knowledge as soon as physicians find an update necessary and new clinical information concerning the choice task is available (Zikos & DeLellis, 2018). A necessary follow-up of the training step is a model validation process (Zikos & DeLellis, 2018). The validation process establishes the extent to which the model will generate choice recommendations that are contextually valid and, consequently, clinically useful.

Most studies on CDSS design validate the performance of the CDSS in terms of a set of classifier performance metrics borrowed from Machine Learning (ML) (Kong, Xu, & Yang, 2008). Examples are studies of Anooj (2012); Gultepe et al. (2014); Lakshmanaprabu et al. (2019); Wu et al. (2020); Ravikumar et al. (2018); Saqlain et al. (2019); Velickovski et al. (2014); Waghlikar et al. (2013); Zikos and DeLellis (2018). The reason for the use of the ML metrics is that CDSSs commonly have a similar goal as ML models: predicting the suited treatment class for a patient. Moreover, most CDSSs are based on ML techniques (see section 1.1). The performance metrics that the CDSS validation in these CDSS studies involve have the following definitions(Tharwat, 2020):

- The accuracy: The ratio between the correctly classified samples to the total number of samples as follows.
- The confusion matrix, containing:
 - True Positives: The choice tasks for which the model correctly predicts the positive class.
 - True Negatives: The choice tasks for which the model correctly predicts the negative class.
 - False Positives: The choice tasks for which the model incorrectly predicts the positive class.

- False Negatives: The choice tasks for which the model incorrectly predicts the negative class.
- Metrics that can be calculated from the Confusion Matrix:
 - Sensitivity (True Positive Rate): The ratio of positives that are correctly identified as positive samples of the total number of positive samples.
 - Specificity (True Negative Rate): The ratio of negative samples that are correctly identified as negative of the total number of negative samples.
 - False Positive Rate: The ratio between the number of negatives incorrectly identified as positive and the total number of negative samples.
 - False Negative Rate: The ratio between the number of positives incorrectly identified as negative and the total number of positive samples.
- Matthews correlation coefficient: The correlation between the observed and predicted classifications.

To conclude, the model update engine consists of a training component that generates an updated model and a validation component that assesses the performance of that updated model in terms of ML-based metrics (Zikos & DeLellis, 2018).

Model quality monitor

A monitor component continuously evaluates the recommendation generator model of the CDSS on any loss of accuracy in a changing context (Gago & Santos, 2008; Velickovski et al., 2014). To this end, it assesses the correctness of the CDSS’s recommendations for new choice tasks (Velickovski et al., 2014). If the number of incorrect recommendations exceeds the level that physicians perceive as unacceptable, the monitor will alert that the performance of the model has decreased significantly. Moreover, the monitor will signal that the CDSS should re-train the recommendation model with the novel information in the adaptive knowledge base. By providing this functionality, the ongoing monitoring of a CDSS is essential in ensuring that the CDSS operates properly and trustworthily (Chen et al., 2007).

CDSS architectures explicitly model a monitor as a system function because this is more effective than trusting physicians with performance assessment activities (Garg et al., 2005; Kawamoto, Houlihan, Balas, & Lobach, 2005). Smith, Geddes, and Beatty (2009) even argue that when designers assume that humans are capable of performing monitoring and updating activities, this results in practically problematic CDSS designs. The incompetency of humans to monitor and maintain the performance of CDSSs stems from the fact that humans, even when highly motivated, are not good at sustained attention tasks (Mackworth et al., 1950; Meister & Enderwick, 2001). Moreover, the assignment of monitoring responsibilities to humans requires the physicians to actively execute an additional step (Garg et al., 2005; Kawamoto et al., 2005). The often occupied physicians easily overlook this step. In short, each CDSS must include mechanisms that detect a potential performance decrease and automatically triggers the CDSS or the physician to act in line with this decrease (Michalewicz, Schmidt, Michalewicz, & Chiriac, 2006). As a result, the CDSS can generate recommendations at the acceptable performance level.

Overview of the main intelligent internal software components

Figure 2.1 visualizes the intelligent internal software components that subsection 2.1.1 suggests. The design details regarding these components depend on the technological ground of the CDSS. For example, knowledge-based systems (KBS) generate choice recommendations by following a set of if-then-else rules (see section 1.1). Accordingly, the update process of a KBS with novel information requires reformulating these rules. On the contrary, non-knowledge-based systems (NKBS) operate data mining and ML techniques — for instance, artificial neural networks or genetic algorithms (see section 1.1). Using the data mining and ML techniques, the CDSS processes new historical data in the knowledge base and finds the most recent decision-making patterns in that data to update (El-Sappagh & El-Masri, 2014). The exact design of the updating engine varies depending on the ML technique that the NKBS employs. An example of a particular updating solution is the Intensive Care Unit CDSS designed by Kolter and Maloof (2007) that operates the classifier updating mechanism suggested by Gago and Santos (2008). Another example of an NKBS updating solution is the Temporal Difference Learning method to ensure a CDSS based

on Artificial Neural Networks can adapt to environmental changes suggested by Baba and Suto (2000).

Because the solutions that literature suggests for dynamic CDSS architectures strongly relate to the technological ground of a CDSS, the designer of a dynamic BAIT-based CDSS architecture cannot directly copy the existing components. Instead, DCM theories, practices, and characteristics delimit the design of these components in the context of a dynamic BAIT-based CDSS. Section 2.2 describes the further investigation of how DCM theories, practices, and characteristics shape the design of the components in the context of a BAIT-based CDSS.

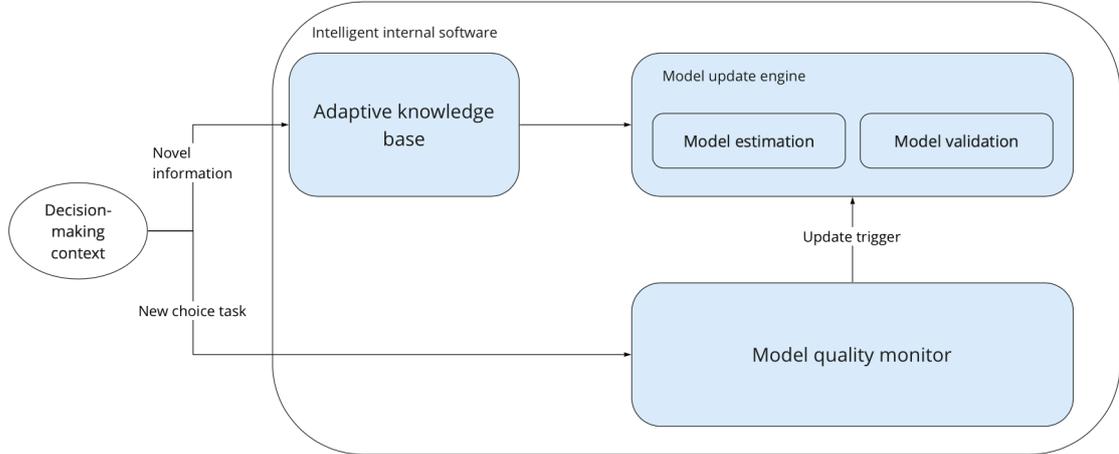


Figure 2.1: Conceptual overview of the intelligent internal software components of a dynamic CDSS.

2.1.2 Human-Computer Interaction

The Human-Computer Interaction (HCI) component of a CDSS takes care of both the feedback provision to inform physicians and the commands that physicians give to supervise the CDSS (Fryszak, 2016). Physicians mainly use CDSSs to receive a recommendation on a choice task they face (Ltifi, Ayed, Kolski, & Alimi, 2009). According to Angehrn and Lüthi (1990) and Ltifi et al. (2009), a CDSS thereby behaves like a cooperative human consultant who supports a physician with a better understanding of a complex choice task. The concept of human supervisory control best defines this role of a CDSS. Human supervisory control is the process in which a human operator intermittently interacts with a system while receiving feedback from and providing commands to a specific process that is part of that system (Cummings, 2006). Angehrn and Lüthi (1990) even state that the primary goal of a CDSS is to enable physicians to interactively explore and analyse decision situations in a way that is compatible with a physician’s thinking.

Because of its dominant role, the HCI component forms an essential part of the CDSS throughout the complete decision-making process (Eapen, 2021; Horsky et al., 2012; Ltifi et al., 2009). Therefore, Cummings (2006) argues that a CDSS design needs to incorporate a physician as a system component instead of as a peripheral entity. By doing so, the designer assumes that a physician’s performance during the interaction with the CDSS heavily influences the CDSS performance. Therefore, the HCI of a CDSS should control the behaviour of a physician (Fryszak, 2016).

Designing for HCI is not straightforward. The existence of different approaches to HCI design even gives rise to a classification of CDSS types (Horsky et al., 2012). In essence, two axes define the main HCI design considerations: the level of automation of a CDSS and the design of the information a CDSS shares.

The level of automation

The level of automation of a CDSS determines the distribution of updating tasks over a CDSS and a physician. As a result, this level of automation shapes the extent to which a CDSS involves a physician in the updating tasks (Lajnef, Ayed, & Kolski, 2005; Ltifi et al., 2009). There are three manners to divide a CDSS updating task over a CDSS and a physician: physicians can fully carry out a task (manual task), a CDSS can fully carry out a task (automatic task) or the task can involve both a physician and a CDSS (interactive task) (Cummings, 2006; El-Sappagh & El-Masri,

2014; Ltifi et al., 2009). The more the accomplishment of a task relies on the skill of a CDSS, the more imbalanced the division of the task is, and the more automatic a CDSS design is. Billings (2018) and Parasuraman and Riley (1997) argue that finding the appropriate level of automation is the primary consideration in the design of CDSSs. Therefore, the division of tasks over a CDSS and a physician is critical in the design of efficient and effective CDSS architectures (Cummings, 2006).

Due to the variety of healthcare contexts (Eapen, 2021), the desired balance of human supervisory control is likely to be context-dependent. Highly automated CDSSs may be preferred in the context of instance Intensive Care Units (ICUs), because these environments are highly dynamic, and physicians have little time to control the maintenance of the CDSS (Gago & Santos, 2008). In other contexts, the restriction of human supervision forms a barrier for the uptake of the CDSS. For example, because the trust in the CDSS is low (Asokan & Asokan, 2015; Grossglauser & Saner, 2014). However, although the purpose of CDSS applications often is to reduce physicians' error and workload, the implementation of highly automated CDSSs that are not entirely reliable will generate new errors in the decision-making context (Cummings, 2006). Therefore, an effective HCI design requires an understanding of the likelihood of errors caused by a physician and a CDSS (Yun et al., 2021). In addition, a designer should consider the degree to which feedback of a CDSS can mitigate a physician's error in a particular task (Yun et al., 2021). By sending specific information to a physician, a CDSS has the power to direct the physician towards particular actions (Khalifa, 2014). The following section covers the design of the information a CDSS presents to a physician.

The design of information

The encouragement of particular behaviour requires the intentional design of the information a CDSS presents to a physician (Djamasbi & Loiacono, 2008; Khalifa, 2014; Silver, 1991). Literature distinguishes four design considerations related to the design of the information that a CDSS provides: the type of information necessary for a specific human action, the content, the timing, and the form of presentation (Eapen, 2021; Frysak, 2016; Khalifa, 2014). Examples of information types in the context of a CDSS are outcome information (OI), task information (TI), cognitive information (CI), or functional validity information (FVI) (Balzer, Doherty, et al., 1989). For example, a CDSS may provide a response or promote insight into the state of the CDSS (Frysak, 2016). The four considerations determine what information is relevant for physicians and how well the information enables physicians to act timely and appropriately in response to the CDSS's feedback (Horsky et al., 2012). Because this research does not focus on interface design, the presentation of information is out of scope.

To conclude, the level of automation and the information a CDSS presents together shape the HCI design of a CDSS. A relation exists between these two aspects (Figure 2.2). The level of automation determines the extent to which the CDSS and the physician interact. If the level of automation is low, a CDSS's updating performance relies heavier on physicians than on the CDSS. Consequently, it becomes more important that the information a CDSS presents drives the correct human behaviour. For instance, by designing an alert that triggers physicians to execute the tasks necessary to maintain the CDSS performance. However, when the level of automation increases, a CDSS's software components regulate the updating performance. In that case, the effect of the information a CDSS presents on a CDSS's performance is negligible.

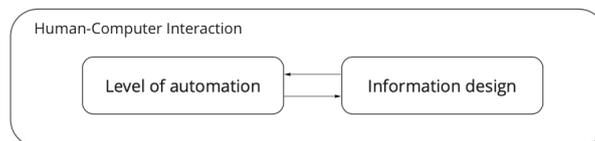


Figure 2.2: Human-Computer Interaction components of a dynamic CDSS.

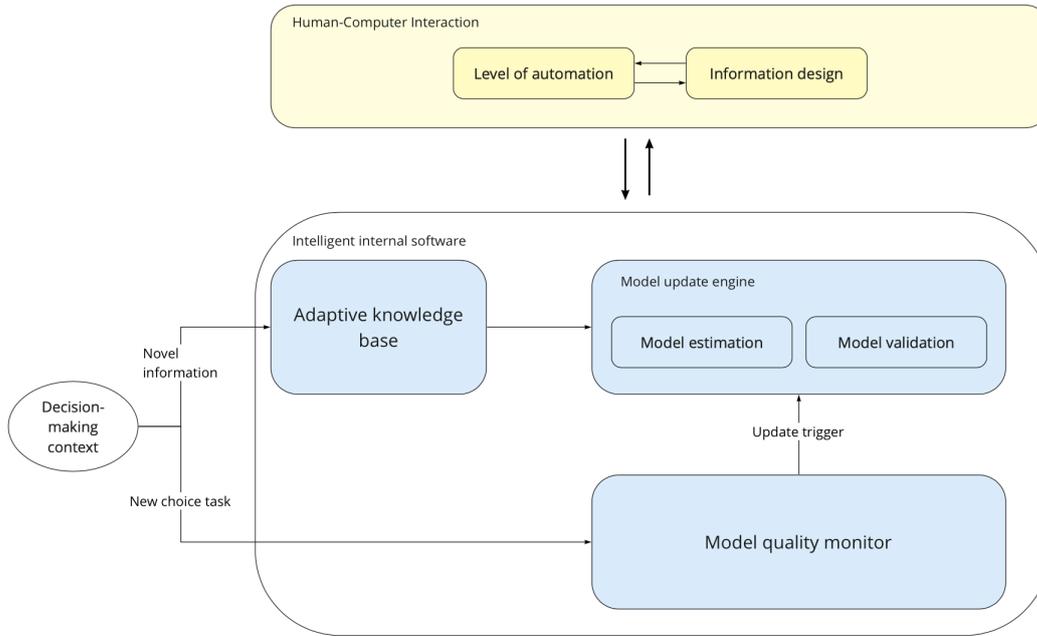


Figure 2.3: Conceptual overview of the main components of a dynamic CDSS.

2.2 The design space set by Discrete Choice Modeling

In the context of a BAIT-based CDSS, DCM theories, practices, and characteristics delimit the design space of the main components that section 2.1 distinguishes. This section describes how these DCM theories, practices, and characteristics shape the design of these components. Figure 2.7 summarizes the findings by giving a conceptual presentation of the design space of a dynamic BAIT-based CDSS architecture at the end of this section.

2.2.1 Adaptive knowledge base

As subsection 1.2.2 explains, DCM choice models are based on choice information (M. Ben-Akiva et al., 1997). Therefore, the adaptive knowledge base in the context of a BAIT-based CDSS is referred to as an adaptive choice base. Choices typically come from two sources: stated preferences (SP's) and revealed preferences (RP's) (Boxall, Adamowicz, Swait, Williams, & Louviere, 1996; Helveston et al., 2018). This report refers to SP's as experiment choices (see subsection 1.2.2), and to RP's as real-life choices.

Experiment choices. An experiment choice is a physician's answer to a hypothetical choice scenario (see subsection 1.2.2). Experiment choices represent the choices physicians state they would make when facing the presented scenario (J. Louviere & Timmermans, 1990). Studies on experiment choice-based models often aim to infer preferences regarding non-existing alternatives (Haider, 2002). For instance, a technological innovation that is not on the market yet. By doing so, researchers desire to test the market potential of these hypothetical alternatives with non-existing attributes. In the context of a BAIT-based CDSS, experiment choices only contain real-life (non-hypothetical) choice attributes and value ranges. An example of a choice attribute in a choice model on the ICU uptake is a patient's BMI (Body Mass Index). Although the choice scenarios only contain real-life considerations, the experiment choices used in a BAIT-based CDSS are still hypothetical. The choices are part of a controlled choice experiment (see subsection 1.2.2) and contain value combinations for the choice attributes that are theoretically possible but very rare in real life. For instance, the scenario in which a patient with an extremely high BMI has very healthy scores on other choice attributes.

Real-life choices. A real-life choice is a choice of a physician about clinical treatment for an actual patient expressed in a real-life situation (Lavasani et al., 2017). Because real-life choices are the result of actual behaviour, real-life choice-based models can cover for the experiment choice-based choice models that lack actual restraints (Qiao, Huang, Yang, Zhang, & Chen, 2016). A downside DCM literature mentions is that real-life choices cannot capture hypothetical elements

and are therefore limited to the characteristics of existing alternatives (Lavasani et al., 2017). However, this is not problematic for a dynamic BAIT-based CDSS. A BAIT-based CDSS aims to model the decision-making strategy that physicians follow to choose between present alternatives.

The estimation of the choice recommendation generator model in the context of a static BAIT-based CDSS only incorporates experiment choices (see subsection 1.2.2). Therefore, the choice base of a static BAIT-based CDSS contains experiment choices. Over time, experiment choices may give an inaccurate representation of the decision-making context. In that case, a new choice experiment might be necessary. The estimation of a choice model should, however, never include a combination of experiment choices collected during different choice experiments. The designer of a choice experiment controls the correlation between choice attributes to maximize the amount of information gained from the choice experiment (Boyce & Williams, 2015). As a result, the choice scenarios that are part of the same choice experiment are interrelated. Instead of combining choices collected during various choice experiments, the adaptive choice base should replace a experiment choice set with a new set.

Real-life choices can also inform a dynamic BAIT-based CDSS about novel developments in the decision-making context. To be able to feed the CDSS with new real-life choices over time, the choice base should store real-life choices. The static version of the BAIT-based CDSS already allows physicians to enter real-life choices into the CDSS. The CDSS processes this information to generate a recommendation on the choice task that the physician entering the real-life choice faces. The BAIT-based CDSS can only process choices that have the same format as the experiment choices used for the estimation of the initial recommendation generator model subsection 1.2.2. As the previous paragraph explains, physicians can enter their choices in this format because the experiment choice format only contains real-life choice attributes.

The storage of real-life choices, however, complicates the design of an adaptive choice base. In contrast to experiment choices, real-life choices may be incomplete because data about patients may be unknown or unclear (M. Ben-Akiva et al., 1997). Moreover, the complexity of a choice task may be high in real life. This high complexity can lead to a cognitive burden. To cope with complex choices, physicians may ignore particular choice attributes (Hensher, Rose, & Greene, 2005). As a result, physicians might solve a real-life choice without considering all the choice attributes a CDSS's recommendation generator model captures.

To conclude, the adaptive choice base of a dynamic BAIT-based CDSS deals with the storage of experiment choices and potentially incomplete real-life choices that physicians enter into the CDSS over time. Additionally, to ensure the set of experiment choices is up-to-date, the choice base should replace these experiment choices with a novel set. Figure 2.4 provides a visual representation of the adaptive choice base.

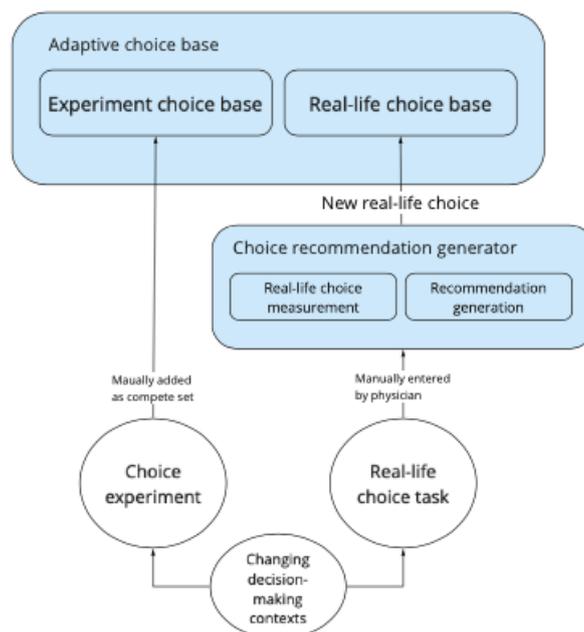


Figure 2.4: Adaptive Choice Base in the context of a dynamic BAIT-based CDSS.

2.2.2 Model update engine

The model update engine trains and validates new choice recommendation generator models. Figure 2.5 at the end of this section, gives a conceptual representation of the model update engine in the context of a dynamic BAIT-based CDSS.

Model training

The training of a BAIT-based CDSS distinguishes it from other CDSSs. As subsection 1.2.2 explains, the recommendation generator model of a BAIT-based CDSS is a choice model that consists of parameters for all choice attributes. The parameters are estimated based on choices made by physicians and represent the relative importance that the physicians assign to the choice attributes. Because a choice model consists of estimated parameters, the model training is referred to as model estimation in the context of a BAIT-based CDSS.

The assumption that the parameters of a choice model are not stable but change over time is not new (M. E. Ben-Akiva, McFadden, Train, et al., 2019; Danaf et al., 2019; Song, Danaf, Atasoy, & Ben-Akiva, 2018). Lachaab et al. (2006) state that if the parameters of a choice model are not attuned to the changing preferences, this will result in misleading outcomes. The parameters of choice models should vary over time to reflect physician’s changing decision-making behaviour (Lachaab et al., 2006).

Finding new parameters requires re-estimating the choice model with novel choice information (Danaf et al., 2019; Siddarth et al., 1995). When the knowledge relevant to a choice task changes in a particular decision-making context, physicians in that context will change their decision-making strategy accordingly. As a result, the experiment and real-life choices of physicians will reflect this developed strategy. When informing the model estimation with new choices over time, the parameters for the choice attributes will change towards the developed strategy in the decision-making context. As subsection 2.2.1 describes, either experiment choices, real-life choices, or a combination of both sources can inform the estimation of a choice model. The following paragraphs describe the differences between the resulting choice models.

Choice models based on experiment choices. Although experiment choices originate from a controlled environment, the data from choice experiments have inherent response biases and significant random errors. Physicians’ decision-making strategy prevalent during a controlled experiment may differ from the strategy used in real life (Morikawa, 1989). According to Carson and Groves (2007), the context in which a choice is made influences the decision-making behaviour of an individual. Even the presentation format of a choice scenario can trigger particular decision-making behaviour (Carson & Groves, 2007; Ding, Grewal, & Liechty, 2005). Therefore, experiment choice-based models may represent the decision-making behaviour that does not represent real-life behaviour.

Choice models based on real-life choices. Choice models based on real-life choices represent real-life decision-making behaviour. A downside is that these choice models often lead to high collinearity between two or more choice attributes in a real-life situation (Brownstone, Bunch, & Train, 2000). High collinearity leads to extensive standard errors in the estimated parameters and to small t-statics (Lavasani et al., 2017). Both relate to inaccurate model estimations (Lavasani et al., 2017). An example of collinearity is the relation between a high age and inadequately functioning organs. In addition, an endogeneity problem can arise when using real-life choices (Helveston et al., 2018). This endogeneity problem leads to biased parameters. In the case of endogeneity, attributes that the choice model does not capture influence a physician’s choice (Helveston et al., 2018). As a result, these unobserved attributes will correlate with attributes that the model does capture (Helveston et al., 2018). The unobserved choice attributes that influence choices are referred to as “noise” and represent the impact of omitted attributes on the choice model. The endogeneity problem is more likely to occur when the number of attributes in a choice situation is large, which is the case for the complex choices of physicians.

When the estimation of the recommendation generator model incorporates real-life choices, the risk arises that the model estimation incorporates choices that the CDSS’s recommendations influenced. While entering a real-life choice, a physician might adjust the choice according to the recommendation that the CDSS provides. It is yet unclear whether physicians or BAIT-based CDSS providers perceive this influence as an issue affecting the trustworthiness of a dynamic BAIT-based CDSS. Moreover, it is yet unclear what solutions are available to mitigate this risk in the context of a dynamic BAIT-based CDSS. Table 2.1 gives an overview of the characteristics of experiment choice-based models and real-life choice-based models in the context of a BAIT-based CDSS.

Table 2.1: *The differences between choice models based on experiment and real-life choices (Helveston et al., 2018; Sanko, 2001).*

	Experiment choice-based model	Real-life choice-based model
Information	Expression concerning a hypothetical scenario.	Expression in real life, result of actual behaviour.
Advantages	<ul style="list-style-type: none"> • Controlled experiment. • Can include information on products and choice attributes that do not exist in the healthcare context, but not expected to be relevant for decision support tool in healthcare. • No measurement error. 	<ul style="list-style-type: none"> • Reflects choices made in real life. • Extensibility of the range of choice attribute values.
Disadvantages	<ul style="list-style-type: none"> • Potential difference in experiment versus decision in real life. 	<ul style="list-style-type: none"> • Incomplete data • Potential for omitted variable bias. • Measurement error. • Collinearity among explanatory variables. • No information on alternatives and choice attributes that are not present in real life (yet), but not expected to be relevant for decision support tool in healthcare. • Incomplete patient information. • Correlated errors among choices.

Choice models based on experiment and real-life choices. Given the advantages and disadvantages of experiment choices and real-life choices as source for the estimation of a choice model, DCM literature does not agree on the preferred source (Swait, Louviere, & Williams, 1994; Helveston et al., 2018). Studies commonly assume that choice models based on real-life choices have superior predictive ability compared to experiment choice-based models. However, other studies emphasize that experiment choice-based models can capture changes in the set of choice attributes (Swait & Louviere, 1993). For instance, when a new choice attribute has to be added or removed from the choice model.

Because both have complementary characteristics, DCM research started to combine both sources in 1990 (M. Ben-Akiva & Morikawa, 1990). DCM literature commonly supports that real-life choices can help “ground” experiment choice-based models in reality (Axsen, Mountain, & Jaccard, 2009; M. Ben-Akiva & Morikawa, 1990; Bhat & Castelar, 2002; Birol, Kontoleon, & Smale, 2006; Brownstone et al., 2000; Brownstone & Small, 2005; Dissanayake & Morikawa, 2003; Feit, Beltramo, & Feinberg, 2010; J. J. Louviere et al., 1999). A joint model estimation allows to slightly adjust the parameters of an experiment choice-based model with real-life choices. These real-life choices represent the contemporary decision-making strategy of physicians. As such, a BAIT-based CDSS can capture contextual change by incorporating real-life choices over time.

However, the combination of both sources in the context of a dynamic BAIT-based CDSS is not straightforward. First of all, possible technological preconditions on estimating a joint choice model for decision support are yet unspecified. Besides, the combination of the two data sources gives rise to an implication. As the previous paragraph mentions, the collection of real-life choices happens under real-life circumstances. Consequently, the contextual “noise” in a real-life setting may influence physicians’ choices. The error term of a choice model captures this noise affecting a physician’s choice. This real-life noise is absent in a controlled experimental setting. As a result, the variance of the error term of an experiment choice-based model and a real-life choice-based model cannot be assumed equal (Beck, Fifer, & Rose, 2016; M. Ben-Akiva & Morikawa, 1990; Swait & Louviere, 1993). However, it is unclear if the differences between the data sources are significant in healthcare decision-making contexts, whether this varies among different healthcare contexts, and whether this is problematic for qualitative decision support.

Model validation

The goal of the model validation is to assess whether a newly estimated recommendation generator model is clinically useful for the contemporary context. To validate the recommendation generator model, a CDSS should contain a validation component that assesses the performance of a new recommendation generator model on test choices every time a new model is estimated (Velickovski et al., 2014; Zikos & DeLellis, 2018). The technological preconditions on the validation of a choice model for decision support are yet unspecified. On the contrary, suggestions regarding validation metrics are in place. Section 2.1.1 presents Machine Learning (ML) metrics commonly used for the performance assessment of CDSSs. Because the architecture design is focused on a CDSS and not on a choice model for behavioral analysis, these metrics form admissible validation metrics for choice models that aim to predict useful recommendations like a dynamic BAIT-based CDSS (Fan, Lin, & Tang, 2017; Franses, 2000).

Since the BAIT approach deviates from the approach of ML classifiers, the performance assessment of a BAIT-based CDSS cannot directly copy ML performance metrics. A formal classifier as specified in ML maps each instance to either a positive or a negative class (Kotsiantis, Zaharakis, & Pintelas, 2006). ML classification models have a deterministic ground truth. This ground truth implies that there is always an objective answer about the class membership. As a result, the CDSS can compare the predicted class to the real class. In the context of a BAIT-based CDSS, this discrete ground truth is absent. Instead, there is a probabilistic ground truth. If a BAIT-based CDSS gives an 80% recommendation, the CDSS states that eight out of ten physicians would vote against treatment. If a physician decides not to proceed with treatment despite a positive recommendation, this does not mean that the recommendation of a BAIT-based CDSS is incorrect. It rather means that the physician holds a minority view.

This subtlety has two implications for the performance assessment metrics in the context of a dynamic BAIT-based CDSSs. First of all, the metrics should get another reference to avoid confusion. Rather than being right or wrong, the recommendation can either correspond with the final choice of the physician or not. As a result, the metric names are as follows:

- Accuracy: Correspondence
- The Confusion Matrix
 - True Positives: Corresponding Positives
 - True Negatives: Corresponding Negatives
 - False Positives: Conflicting Positives
 - False Negatives: Conflicting Negatives
- Main metrics that can be calculated from the Confusion Matrix:
 - Sensitivity (True Positive Rate): Correspondence sensitivity
 - Specificity (True Negative Rate): Disagreement specificity
- Matthew correlation coefficient: Not renamed, because of its unique reference.

Second, the performance validation needs a threshold defining when a recommendation is a vote in favour or against a specific treatment. This threshold represents the minimal majority an end user needs to see perceives as convincing. The classification of recommendations with the threshold allows assessing which recommendations were in line with the real-life choice of the physician. By doing so, the threshold value influences the performance assessment: if the value increases, the chance that a recommendation deviates from the physician who decided to proceed with the treatment increases. It is yet unclear whether physicians in different contexts have contrasting opinions regarding the threshold value and what the preferences on this threshold are.

Next to the ML-based metrics, two additional metrics are suitable for the performance assessment in the context of a BAIT-based CDSS. First of all, a metric that is peculiarly related to the technological foundation of a BAIT-based CDSS. As the previous paragraph explains, the recommendation of a BAIT-based CDSS represents the internal agreement within a pool of physicians. If the internal agreement is high, the CDSS recommends with little confidence. Most likely, the recommendation to proceed with the treatment is around 50%. As such, the recommendation is

either a correct or incorrect reflection of the real-life internal agreement. If a BAIT-based CDSS and a physician are both uncertain regarding a particular choice, the CDSS correctly understands the complexity of the choice task. As a result, the difference between the recommendation of a BAIT-based CDSS and the physician’s confidence regarding the specific real-life choice gives insight into this representation power of the CDSS. Accordingly, averaging the differences over a set of real-life choice recommendations provides insight into the performance of a dynamic BAIT-based CDSS over a series of real-life choice tasks. This average is referred to as the “Confidence Representation” in the remainder of the report.

Second, DCM literature proposes a metric that DCM researchers commonly use to validate choice models: the Rho-squared value (de Luca & Cantarella, 2009). The Rho-squared value illustrates how well the estimated parameters of a choice model fit the choice data. The metric value ranges from a model that does not perform better than throwing a dice to a model that forms a perfect fit and becomes deterministic. Moreover, standard DCM diagnostics may also be helpful for the validation of choice models in dynamic contexts. For instance, the size of the changes in parameter estimations and in standard errors associated with these parameter estimations. The errors represent the reliability of the estimated parameters. It is yet unclear which metrics are of interest to physicians or BAIT-based CDSS providers.

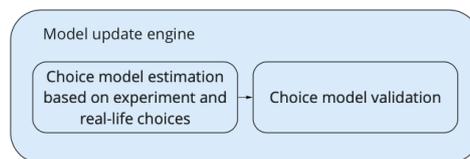


Figure 2.5: Model Update Engine in the context of a dynamic BAIT-based CDSS.

2.2.3 Model quality monitor

The model quality monitor constantly assesses the performance of the CDSS. In the context of a dynamic BAIT-based CDSS, a continuous performance assessment concerns examining the difference between a real-life choice of a physician and the CDSS’s recommendation on this real-life choice (see section 2.2.2). When the recommendation corresponds with the choice made by the physician, the BAIT-based CDSS correctly mimics the decision-making strategy of physicians active in the decision-making context. As Figure 2.6 illustrates, the performance assessment consists of two steps in the context of a dynamic BAIT-based CDSS. The first step the monitor should execute is comparing the CDSS’s recommendation with the majority threshold (see section 2.2.2). This comparison establishes whether the recommendation represents a vote in favour or against the treatment. The second step the monitor should execute is examining whether the recommendation is in line with the physician’s choice.

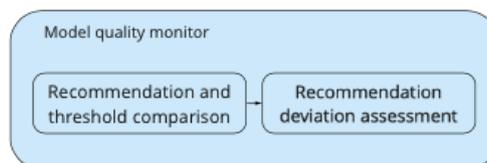


Figure 2.6: Model Quality Monitor in the context of a dynamic BAIT-based CDSS.

As section 2.1.1 explains, the model quality monitor should indicate the need for or even directly trigger an update. When an update is necessary depends on the CDSS performance level that physicians perceive as acceptable. This performance level represents the number of times a dynamic BAIT-based CDSS can generate recommendations that deviate from the physicians’ choices, while assuming that the recommendation generator model does not need an update. In the remainder of this report, this number is referred to as the level of acceptance. The level that physicians find acceptable is yet unknown. However, this level is likely to be context-specific. In addition, it is unknown which specific actions physicians want a dynamic BAIT-based CDSS to undertake when obtaining this level. A physician may wish for the CDSS to push a model

update directly. Another physician may prefer to first further investigate the performance decline. Figure 2.7 summarizes the findings of section 2.2 by giving a conceptual presentation of the design space of a dynamic BAIT-based CDSS architecture.

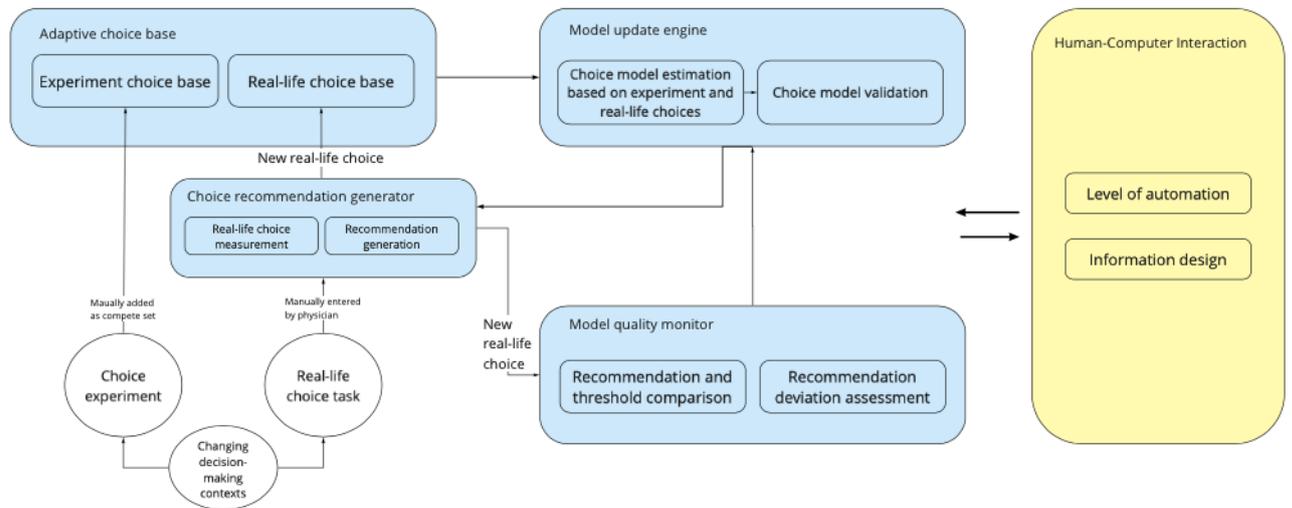


Figure 2.7: Conceptual outline of the design space of a dynamic BAIT-based CDSS architecture.

2.3 Summary chapter 2

The goal of chapter 2 is to present the main components of a dynamic CDSS and describe how Discrete Choice Modeling (DCM) theories, practices, and characteristics shape these components. An architecture that defines a dynamic CDSS commonly describes four main components: an adaptive knowledge base so that the CDSS accepts new information, a model update engine so the recommendation generator model continually bases its recommendations on that new information, a model update monitor so that the CDSS timely detects any decrease in performance in a changing context, and a Human-Computer Interaction (HCI) component. CDSS and CDSS architecture literature give solutions for the design of these components that are technology-dependent. Therefore, additional efforts are necessary to inform the design of these components in the context of a dynamic BAIT-based CDSS architecture. Table 2.2 presents the architecture design space framework summarizing the main components and the design gaps regarding these components. The construction of the framework allows answering the first and second sub-question:

1. **What are the main components of a dynamic CDSS architecture?**
2. **How is the design of the main components of a dynamic CDSS architecture shaped by the theories, practices and characteristics of DCM?**

The header of the architecture design space framework specifies the main components that CDSS and CDSS architecture design literature suggest. The first row defines the elements of each main component. The next row summarizes how DCM theories, practices, and characteristics restrict the design of each component. The final row specifies which unknown aspects need additional research to outline the design of each main component in the context of a dynamic BAIT-based CDSS further.

The expansion of a CDSS with additional functionalities gives rise to design considerations regarding HCI. Consequently, the design of a dynamic BAIT-based CDSS architecture also concerns the design of HCI. In the context of a CDSS, two HCI design aspects are essential. The first aspect is the level of automation. For this research, this level concerns the automation of CDSS updates. The second aspect is the design of the information a CDSS presents to physicians. A direct relation between the two aspects exists: the higher the level of updating automation, the less prevalent the information design becomes. The level of updating automation should suit the preferences in the decision-making context and take into account the likelihood that a physicians or

a CDSS causes an error when performing a particular task. The information design can influence the likelihood that a physician causes an error because the right information design can shape a physician’s behaviour.

DCM theories, practices, and characteristics do not explicitly influence the HCI design. However, gaps regarding the HCI design for a dynamic BAIT-based CDSS exists: the information physicians want to receive, the influential design of the information, and the desired level of updating automation. Depending on the variety in preferences regarding these aspects, the architecture design is challenged with finding the right combination of updating task automation and information provision for multiple contexts.

Table 2.2: *The architecture design space framework.*

Model component:	Adaptive knowledge base	Model update engine	Model quality monitor	HCI component
Sub processes:	Data measurement, Data storage.	Model training, model validation.	Assessment of the acceptability of the recommendations for individual cases.	No components specified, but considerations: level of updating automation and information design in terms of type, content, and timing.
DCM complications:	Need for storage of both experiment and real-life choice data, for replacement of experiment choice base while considering experiment choices as one set, for continuous acceptance of real-life choices, for management of incomplete real-life choices.	Real-life choices might capture “noise”, combination of different choice sources, estimation based on real-life choices a dynamic BAIT-based CDSS influenced, need for adjusted performance metrics, for a comparison of recommendation and threshold.	Need for a comparison of recommendations and threshold, for a deviation assessment between recommendations and choice physician, for a level of acceptance.	-
Unknowns in context of a dynamic BAIT-based CDSS:	Most reliable way to manage incomplete patient information in a choice.	Preconditions on the execution of an update, the performance metrics of interest, preferred value for majority threshold by physician, and the preference variety, degree to which different variances are expectable, problematic and DCM based suggestions form a suitable solution.	The preferred consequences when performance has declined below the accepted level, and the preference variety, preferred level of acceptance and the preference variety among contexts.	The preferred information, preferred level of updating automation and the preference variety among contexts, preferred information design and the preference variety among contexts.

Chapter 3

Introduction to the action context: The current and desired situation

This chapter presents the requirements of a dynamic BAIT-based Clinical Decision Support System (CDSS) architecture for Council. The architecture design space framework that chapter 2 presents provides insight into the CDSS components and design considerations that are relevant when designing a dynamic BAIT-based CDSS architecture. However, the architecture design space framework also highlights the aspects of the architecture design that are yet unknown. The move towards the action context enables the collection of practically relevant and in-depth knowledge necessary to answer these unknown aspects and further outline the architecture design. Section 3.1 gives an introduction to the problem that Council experiences. This introduction forms the starting point of identifying the situation Council ultimately desires. Section 3.2 describes this desired situation in terms of architecture requirements. The chapter concludes with a summary in section 3.3. The findings that this chapter presents provide the answer to the following question:

3. What are the requirements for a CDSS architecture of a dynamic BAIT-based CDSS?

3.1 Introduction to the current action context

This section introduces the action context as it is. This section first introduces the key stakeholders that are central in the remainder of this report (subsection 3.1.1). Next, this section explicates the problem Council currently faces (subsection 3.1.2).

3.1.1 Introduction to Council and the key stakeholders

Council is a spin-off of the University of Technology Delft. Council is a CDSS provider and has been developing BAIT to offer transparent decision support since April 2020. As a CDSS provider, Council mainly employs CDSS developers who serve physicians with a BAIT-based CDSS. Because Council's primary goal is to develop CDSSs, this research does not distinguish between needs of Council's CDSS developers and other representatives of Council. The developers customize a BAIT-based CDSS for a particular choice task the physicians in a particular decision-making context face regularly (see subsection 1.2.2). An example is a CDSS that assists the choice task regarding the uptake of a patient in the Intensive Care Unit (ICU).

All physicians active in the same decision-making context can consult the BAIT-based CDSS. Therefore, each BAIT-based CDSS has a group of physicians as clinical end users. Council distinguishes two types of clinical end users in each group of physicians using the same CDSS. The first type is the average end user who consults the CDSS for choice recommendations. The second type also consults the BAIT-based CDSS but is also the CDSS product owner. This type of end user is responsible for all final decisions concerning the CDSS and is the first point of contact for Council. Therefore, this report refers to the second type as the main end user.

Both types of end users apply a BAIT-based CDSS with the same goal: receiving decision support. Besides, end users active in the same context will use the same BAIT-based CDSS and have to agree on its characteristics and functionalities. Therefore, this research does not distinguish between the preferences of the main end user and average end users operating in the same decision-making context. The remainder of this report refers to both types with the term "clinical end user".

Although this report does not distinguish between the preferences of the different end user types, expressing the distinction between these user types is necessary to understand the solution this research gives to Council’s problem. Figure 3.1 gives an overview of the two key stakeholders that this section introduced: Council and the end user.

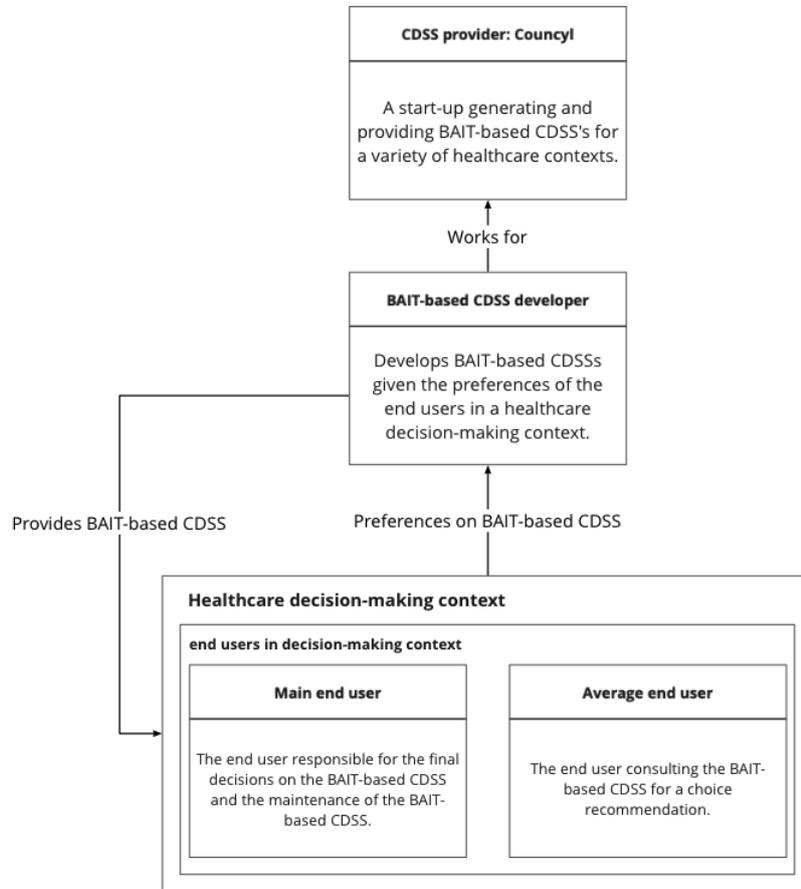


Figure 3.1: Stakeholders of the action context: Council and the end users.

3.1.2 Introduction to the problem experienced by Council

As section 1.3 explains, the current version of the BAIT-based CDSS is static and only generates choice recommendations based on clinical decision-making knowledge available at the moment of development (for an architecture of the current static BAIT-based CDSS, see Appendix D). Although the static BAIT-based CDSS has promising features for qualitative decision support in healthcare contexts, the CDSS is inadequate for the application in dynamic decision-making contexts. As a result, the BAIT-based CDSS that Council currently offers has shortcomings. According to Council, this inadequacy to match with the dynamic context can be traced down to the following missing features:

- The BAIT-based CDSS does not retain its accuracy over time, because there is no established updating mechanism that lets a choice recommendation generator model incorporate developments in the decision-making context.
- The BAIT-based CDSS cannot estimate a choice recommendation generator model based on a combination of experiment choices and real-life choices, which are both collected under different conditions.
- The BAIT-based CDSS cannot deal with the characteristics of choices collected in a real-life setting. For instance, real-life choices may contain missing values.
- The BAIT-based CDSS does not assess the performance of a choice recommendation generator model in a changing context on a continuous basis.

Council needs to transform the static BAIT-based CDSSs so that it matches the dynamic contexts and retains its value to healthcare decision-making contexts over time. To this end, the current static BAIT-based CDSS needs an expansion with additional features. However, this expansion requires addressing the existing shortcomings. Doing so is challenging for three reasons:

- There are no prefabricated solutions or starting points at hand because of the novelty of the application of Discrete Choice Modeling (DCM) for CDSS development. Council does not know what solutions can cover the missing features, what the risks of these solutions are, and how a CDSS can combine these solutions effectively.
- The healthcare contexts Council aims to serve with a BAIT-based CDSS heavily vary (Eapen, 2021). Possible differences between contexts are the speed with which the decision-context changes, the frequency with which the choice task occurs, and the interaction clinical end users want with the CDSS. Council does not know which adjustments are necessary to satisfy the preferences of healthcare clients best.
- Because healthcare contexts are dynamic, the preferences of healthcare clients are likely to change over time. Council does not know how to take into account the changing preferences, while covering the features that the static BAIT-based CDSS lacks.

To serve clinical end users with accurate choice recommendations during ongoing CDSS use, the CDSS developers of Council need a CDSS architecture that shows how to facilitate a fit with the dynamic decision-making context. This fit must align with the varying and changing preferences of clinical end users in different decision-making contexts. As a result, the architecture should form a guiding tool for developing a dynamic BAIT-based CDSS for a particular healthcare decision-making context.

3.2 Desired situation: Architecture requirements

This section presents the final set of architecture requirements that outline the architecture as is desired by Council. First, subsection 3.2.1 gives an introduction to the concept of architecture requirements. This section also presents the categories of architecture requirements that this report distinguishes. Section 3.2.2 describes the methodology this research followed to identify the architecture requirements. Section 3.2.3 presents the final set of architecture requirements.

3.2.1 The classification of architecture requirements

A requirement is a property of an artefact deemed desirable by stakeholders of the artefact (Johannesson & Perjons, 2014). As such, a set of requirements describes what an artefact should be and do from the point of view of the stakeholders (Proper & Greefhorst, 2010). Accordingly, requirements guide the design and development of the artefact. A requirement can concern the functions, structure, or environment of an artefact. It can also cover the effects of using the artefact. By doing so, requirements function as selection criteria for choosing the proper artefact design (Di Noia, Mongiello, Nocera, & Straccia, 2019).

The difference between CDSS requirements and architecture requirements

The artefact subject to design in this research is an architecture. Because an architecture describes the structure of a system, the specification of the architecture requirements will affect the functioning of the system (Gong, 2012). Moreover, architectures often function as a tool to communicate and illustrate the features of a system during a system development cycle. As a result, CDSS architecture literature mainly discusses requirements at CDSS level and barely defines requirements at the level of the architecture (Gong, 2012; L. Tabares, Hernandez, & Cabezas, 2016; F. Tabares, Hernandez, & Cabezas, 2017). For example, Hoogervorst (2009) states that architecture requirements are areas of concern related to the development a CDSS or to other aspects that are critical for the development and implementation of that CDSS. Similarly, other studies mention that architectures should incorporate both the functional requirements that cover the functionalities that a CDSS must provide, and the Quality Attributes Requirements (QAR) that cover the qualities a CDSS must meet when delivering the functional requirements (Erder & Pureur, 2015).

Categories of requirements at architecture level

Because this research focuses on the design of an architecture, the requirement identification does not aim to formulate requirements at the level of the CDSS. Nevertheless, the architecture will embed components that realize certain CDSS features. As such, the architecture requirements should not only concern the desired architecture features. Instead, the requirements identification involves translating CDSS features into requirements at the level of the architecture. To guide this translation, this research distinguishes three categories of architecture requirements.

First of all, the architecture is an artefact that Councilyl will use for the development of CDSSs. For the architecture to form a tool that solves Councilyl’s problem, the architecture should meet particular requirements (see section 3.1). This research refers to these requirements as client artefact requirements (CAR).

The second category consists of QARs. The previous paragraph gives a brief introduction to QARs. QARs are requirements concerning the qualities of a CDSS that a single CDSS component cannot realize (Gorton, 2011). Instead, the realization of a QAR involves the complete structure of the CDSS. The architecture of a CDSS describes this structure. Although QARs concern important CDSS qualities, the CDSS architecture design determines the extent to which a CDSS will exhibit particular quality attributes (Erder & Pureur, 2015; Gorton, 2011; O’Brien, Merson, & Bass, 2007). Therefore, QARs ensure that the architecture design guides the development of a CDSS that possesses the desired qualities. Consequently, the architecture should result from design decisions that adhere to the QARs (Erder & Pureur, 2015).

Next to CDSS qualities, the architecture should also ensure that the CDSS provides clinical end users with the desired functionalities. Therefore, this research distinguishes requirements that specify which functionalities an architecture should force at the level of the CDSS. By adhering to these requirements, the architecture directs the development of specific CDSS components. This report refers to these requirements as development guiding requirements (DGR). Figure 3.2 visualizes the requirement classification. For clarification purposes, Table 3.1 defines each type of architecture requirement that this report distinguishes.

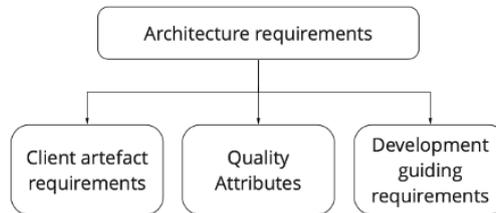


Figure 3.2: Classification of architecture requirements.

Table 3.1: *Definitions of the architecture requirement categories*

Requirement category	Definition
Architecture requirement (AR)	The functional and non-functional requirements that pertain to the architecture of a class of CDSSs (Greefhorst & Proper, 2011).
Client artefact requirements (CAR)	The functional and non-functional architecture requirements that pertain to the architecture as an artefact.
Quality Attribute Requirements (QAR)	The functional and non-functional architecture requirements that address issues of concern to stakeholders of a CDSS and cannot be captured by a single component or part (Gorton, 2011).
Development guiding requirements (DGR)	The functional architecture requirements that specify what CDSS functionalities the architecture should force on a CDSS.

3.2.2 The methodology of the requirement identification

This section describes the methodology of the requirement identification. The section starts with the sources that informed the requirement identification. One of the sources are the interviews

conducted with the key stakeholders. The subsequent sections present the interview structure, the interview participants, the interview setting, and the interview questions.

Sources of the requirement identification

Three sources enabled the formulation of the final set of architecture requirements. As section 1.5.2 explains, the output of six interviews with the key stakeholders formed an initial list of requirements. Appendix A presents a table with practical details on all interviews. The analysis of the interview output involved CDSS design and CDSS architecture design literature. The literature both shaped and complemented the analysis. The literature shaped the interview analysis because it supported the translation of aspects that participants stated into feasible and relevant architecture requirements. The literature complemented the interview analysis because it provided insight into important aspects that participants did not explicitly mention. A potential reason for participants to be incomplete in their reasoning is that the architecture defines an expansion of the static BAIT-based CDSS that is already in place. Participants may overlook aspects already captured by the static CDSS, but are still relevant for the dynamic CDSS. Moreover, the participants form a small selection of all physicians. Accordingly, the interview answers may be biased. Third, interviews are subjective by nature. Literature provides an objective perspective and relaxes the subjectivity of the interview data (Johannesson & Perjons, 2014; McIntosh & Morse, 2015). Appendix B gives the complete description of the interview analysis, and shows how the literature informed the analysis. As a third source, the initial requirements were refined with new insights and lessons learned during the entire research (for the lessons learned that shaped the formulation of the requirements, see chapter 7).

The structure of the interviews

The interviews followed a semi-structured format. During a semi-structured interview, the interviewer elicits information from the interviewee with questions that unfold in a conversational manner. By doing so, the interviewer has the opportunity to explore issues that the interviewee thinks are relevant (Longhurst, 2003). Where structured interviews may result in stale answers, which are undesired when eliciting requirements, semi-structured interviews give the participants space to take the initiative and guide the conversation into unexpected directions (Agarwal & Tanir, 1990; Queirós, Faria, & Almeida, 2017). As such, a semi-structured format enables one to dive deeper into surprising answers (Queirós et al., 2017). A disadvantage is that semi-structured interviews can lead to abstract and off-topic answers (Johannesson & Perjons, 2014). Therefore, participants receive an interview procedure in advance of the interview. Another drawback is that data gained during interviews depends on the perspective, stakes, and interests of the specific person being interviewed (McIntosh & Morse, 2015). As suggested by Checkland and Scholes (1990), the interviews involved different representatives of Council and different types of clinical end users to get a complete overview of the problem and solution expectations. Section 3.2.2 introduces the interviewed participants.

Participants of the interviews: key stakeholders

The selection of the interview participants is vitally important (Longhurst, 2003). As the previous section explains (section 3.2.2, a homogeneous selection of participants will lead to a restricted perspective on the requirements. Therefore, a diverse set of stakeholders has to be involved in the identification of the requirements (Checkland & Scholes, 1990; Shah & Patel, 2016). As section 1.5.2 describes, the interview participants are the key stakeholders. The key stakeholders are Council and the CDSS end users introduced in subsection 3.1.1.

Participants - Council

As Checkland and Scholes (1990) suggests, the interviews involved different representatives of Council. By doing so, the interview output offers a complete overview of the problem perceptions and the expectations regarding the problem solution. The selected representatives all have a high stake in or a strong influence on the architecture design:

- Co-founder and CEO of Council: has a high stake in the architecture design since this participant has to use the architecture to help different healthcare clients with decision-making in dynamic contexts in the future. This participant assists clients in codifying and automating decision-making processes. To this end, this participant will implement the architecture in the existing software.

- Co-founder and scientific advisor of Council: has a strong influence because this representative possesses critical knowledge about DCM and BAIT. Therefore, this representative knows which theoretical considerations are relevant and which design alternatives are feasible in the context of a dynamic BAIT-based CDSS architecture.
- Decision analyst of Council: has a high stake in the architecture design since this participant will use components defined in the architecture to apply a dynamic BAIT-based in various healthcare decision-making contexts.

Participants - clinical end users

For the same reason as the previous paragraph gives, the interviews involved three different types of clinical end users. Two criteria guided the selection of the end users for the interviews. First, the degree of experience the end user has with a BAIT-based CDSS. The degree of experience determines the knowledge of a BAIT-based CDSS. The dynamic BAIT-based CDSS should fit the preferences of all potential end user, including existing and future users. Second, the frequency of the choice task. The frequency determines how many choices that capture new contextual knowledge are available in the context. Therefore, the frequency of the task represents the amount of information that is available for updates of a CDSS. The dynamic BAIT-based CDSS should be useful for all potential choice tasks, not matter how often it occurs. By guiding the selection with these two criteria, the interviews generate a complete perspective on the end users' preferences.

- A clinical end user who is an “advanced” client of Council. The choice task rarely occurs: approximately 20 times a year.
- A clinical end user who is a “new” client of Council. The choice task often occurs: approximately 100 times a year (bi-weekly).
- A clinical end user who is not yet considered a client of Council. The choice task regularly occurs: approximately 50 times a year (weekly).

Interview setting

Each interview took 45 minutes. Due to the COVID situation, the interviews were conducted online via Microsoft Teams and Zoom. The representatives of Council were situated in their working environment, where they will use the architecture. The clinical end users were situated at the healthcare location where they make decisions and use or will use the support of the BAIT-based CDSS.

Interview questions

The goal of the interviews is to identify the architecture requirements. These requirements outline the architecture. The architecture design space framework created in chapter 2 forms an initial outline. This framework presents the main components of a dynamic CDSS and the restrictions that DCM theories, practices, and characteristics pose on these components. Moreover, the architecture design space framework highlights the aspects that are still unclear. Therefore, the architecture design space framework assists in selecting the focus areas of the interview questions. Table 3.2 presents the interview questions for the interviews with representatives of Council. Table 3.3 presents the interview questions for the interviews with end users. Both tables with interview questions contain a column “Rationale/Framework reference”. This column describes why the question is relevant. If the question is directly derived from the architecture design space framework, the column explicitly mentions this as well.

Table 3.2: *Questions and rationale semi-structured interviews Council*

Reference	Question	Rationale/Framework reference
1	What is the goal of this design research project?	Requirement identification should start with the goals of the project and proposed artefact (Zhou, 2004). The goals of the project and the artefact indicate what characteristics and functions the artefact should have.
2	What is the goal of the to be designed CDSS architecture?	Idem.
3	Can the goal be further specified in sub-goals?	To understand the vision of Council regarding the architecture design project and identify more precise business objectives, and while doing so moving from the current to the future state.
4	When is this goal/ are these goals achieved?	To get a clear overview of the features that are of value for a dynamic BAIT-based CDSS architecture. In addition, focusing on achievement frames the discussion in a positive light and forces to focus on the benefits, while making the key stakeholders excited about the project.
5	What should the architecture be capable of doing in order to achieve this goal? Per function: why?	Framework: check if additional features are needed. Cover for potentially missing components of value for Council.
6	What characteristics should the architecture have in order to achieve this goal? Per characteristic: why?	Framework: check if additional features are needed. Cover for potentially missing components of value for Council.
7	With the knowledge and experience regarding how clinical end users use BAIT, would like to suggest additional features and/or functions to what is already discussed?	Framework: check if additional features are needed. Cover for potentially missing components of value for Council.

Table 3.3: *Questions and rationale semi-structured interviews clinical end users*

Reference	Question	Rationale/Framework reference
1	How would your trust in and acceptance of a dynamic BAIT-based CDSS change when the model would incorporate both experimental and real-world choices made by you and your colleagues?	Framework: real-life choice data may contain “noise”, which results from decision-making in the real-world rather than in a controlled environment. It is yet unknown whether end users trust in a dynamic BAIT-based CDSS when it incorporates noise captured by real-life choices.
2	If a dynamic BAIT-based CDSS can be updated with novel choice information, would you prefer that a dynamic BAIT-based CDSS incorporates all choices or just a specific set of choices? For instance, specific choice situations which should be excluded.	Framework: a dynamic BAIT-based CDSS is based on the choices in the choice base. However, because of the variety of contexts (Eapen, 2021), clinical end users may want to specify the choices that influence an estimation and exclude other types of choices. It is yet unclear what preferences regarding the choice selection are and whether they indeed vary.
3	If a BAIT-based CDSS can update, different choices could be differently weighted. Can you explain what you think of the following five options. A dynamic BAIT-based CDSS is always influenced by real-life choices as much as by experiment choices (1), a dynamic BAIT-based CDSS is always less influenced by real-life choices than by choices as from choices made during a choice experiment (2), a dynamic BAIT-based CDSS is always heavier influenced by real-life choices than by experiment choices (3), it depends on the characteristics of the choice how much a dynamic BAIT-based CDSS should be influenced by the choice (4), the end user who made the choice should determine how much a dynamic BAIT-based CDSS should be influenced by a specific real-life choice (5).	Framework: a dynamic BAIT-based CDSS is based on the choices in the choice base. However, because of the variety of contexts (Eapen, 2021), clinical end users may want specific choice types to have a larger influence on the estimation. It is yet unclear what preferences regarding the choice weights are and whether they indeed vary.
4	What would be a reason for you to ensure an update is less influenced by a specific type of choice observation?	Framework: a dynamic BAIT-based CDSS is based on the choice observations in the knowledge base. The reasons for differences in the importance of different type of choice observations are unknown. These could potentially inform the need of additional features, like dealing with situations that are beyond the regular decision-making structure and initially not captured by a dynamic BAIT-based CDSS.
5	How should a dynamic BAIT-based CDSS deal with incomplete patient information of a choice?	Framework: real-life choices may be incomplete. Information on the choice may be missing or unknown. However, it is not yet clear how end users would like the CDSS to deal with missing patient data.
6	Under what circumstances or according to which criteria should a dynamic BAIT-based CDSS be updated?	Framework: the architecture should establish an updating mechanism. However, it is yet unclear whether end users indeed have different preferences regarding the activation of an update. It is of interest whether an update should be available or be restricted by specific preconditions.
7	How should an update of a dynamic BAIT-based CDSS be activated?	Framework: the human-computer interaction can be shaped by the level of updating automation. However, it is yet unclear whether end users indeed have different preferences regarding the level of updating automation regarding the updating tasks of a dynamic BAIT-based CDSS. For instance, whether an update should be activated by an end user or initiated by a dynamic BAIT-based CDSS itself.
8	What information would you like to receive from a dynamic BAIT-based CDSS and when?	Framework: the human-computer interaction can be shaped by the information the CDSS provides to the end user. However, it is yet unknown whether end users want to receive information regarding updates and if end users have different opinions regarding the type, content, presentation and timing information.
9	How should the CDSS deal with the situation in which a clinical end user’s choice deviates from the recommendation of a dynamic BAIT-based CDSS?	Framework: a monitor assessing the performance over time is a key component of a dynamic CDSS. However, it is yet unknown what end users want the CDSS to do when the CDSS notices a performance decline and if end users in different decision-making contexts have varying opinions regarding this percentage.
10	What percentage of the clinical end users behind a dynamic BAIT-based CDSS should vote in favour of the treatment for you to take the recommendation as a serious vote in favour of the treatment?	Framework: a majority threshold is to be defined in case of a dynamic BAIT-based CDSS. However, it is unknown what an appropriate threshold is and if end users in different decision-making contexts have varying preferences regarding this threshold.
11	How many times may the CDSS give a deviating recommendation until you think an update is needed?	Framework: a limit defining when the decline in performance is no acceptable anymore is to be defined in case of a dynamic BAIT-based CDSS. However, it is unknown what an appropriate threshold is and if end users in different decision-making contexts have varying preferences regarding this threshold.

3.2.3 The results of the requirement identification

As subsection 3.2.1 explains, this research distinguishes three types of architecture requirements: the Client Artefact Requirements (CAR), Quality Attribute Requirements (QARs), and Development Guiding Requirements (DGR). Appendix B presents the interview analysis from which the requirements were derived.

Client Artefact Requirements

The CARs define the architecture requirements that pertain to the architecture from Council’s point of view. In essence, Council desires to use the architecture to develop dynamic BAIT-based CDSSs for all potential healthcare end users. Therefore, the architecture should form a guiding tool for Council’s developers.

The architecture should guide the development so that Council can generate a dynamic BAIT-based CDSS for application in a specific healthcare context within four weeks. For a smooth development of a dynamic BAIT-based CDSS, the description should be complete, reliable, and uniform. Complete implies that developers of Council are confident that no significant research efforts will be needed to develop a CDSS that achieves the intended goals. A reliable architecture will eliminate any doubts about the quality of the defined processes. A uniform will guarantee that developers cannot interpret the architecture in multiple ways. As such, uniformity ensures that the architecture does not lead to the development of unintended CDSS designs.

Because healthcare decision-making contexts are heterogeneous (Eapen, 2021), preferences regarding the CDSS will differ among the healthcare clients whom Council serves. For the architecture to describe a CDSS that possesses the abilities and characteristics considered important by all clinical end users, the architecture should guide the development of different forms of a dynamic BAIT-based CDSS. Moreover, user preferences will change over time. The CDSS’s functionalities and qualities the architecture forces on a CDSS should therefore be modifiable over time.

Finally, the architecture forms an expansion of the current static BAIT-based CDSS structure. Hence, the architecture should align with the organizational procedures that Council installed for the operation of the static BAIT-based CDSS. An important aspect is adhering to privacy statements and protocols of Council, which adhere to the General Data Protection Regulation (GDPR). On the next page, Table 3.4 lists the resulting CARs.

Quality Attribute Requirements

QARs drive how the architecture should describe the components and functionalities of the CDSS (Gorton, 2011; O’Brien et al., 2007). The interviews and literature review showed that four quality attributes are relevant concerning a dynamic BAIT-based CDSS: trustability, ease of use, supportability, and service availability. The subsequent paragraphs give a further breakdown of and explanation of these attributes. Table 3.4 provides an overview of all QARs.

Trustability

The uptake and usage of Artificial Intelligence (AI) technologies, like BAIT-based CDSSs, strongly correlates to the trust in these technologies (Gefen, Karahanna, & Straub, 2003; Siau & Wang, 2018). Trustworthy AI forms a prerequisite for a responsible and ethical application of AI tools. Especially for the implementation in sensitive areas like healthcare contexts (Ten Broeke et al., 2021). For the development of initial trust in the tool, the processes of a dynamic BAIT-based CDSS need to be explainable, and trialable (Siau & Wang, 2018). Explainability concerns the transparency of the components and outcomes of the AI tool and the ability to justify these components and outcomes (Siau & Wang, 2018). Transparent implies that a dynamic BAIT-based CDSS should make its internal change visible so that end users can track how the CDSS’s recommendation model evolves. However, if the end user cannot comprehend the inner workings of the dynamic CDSS, this transparency is pointless. The more complex an AI tool becomes, the less the tool will be capable of self-explanation in an intuitive way (Hacker, Krestel, Grundmann, & Naumann, 2020). Unexplainable AI tools are problematic in the healthcare sector where choices involve patients’ well-being: ethical issues will occur if clinical end users cannot understand or validate the CDSS components and outcomes (Alexander, 2006). As such, a dynamic BAIT-based CDSS must be explainable so that the components and outcomes are understandable for every clinical end user (Siau & Wang, 2018). Trialability provides clinical end user with the opportunity to try out components by investigating the effects of particular actions and undo these actions if they are not desired. In the context of a dynamic BAIT-based CDSS, clinical end users should

Table 3.4: *Client Artefact Requirements*

Reference	Rationale	Requirement
CAR1	Enable smooth development, Reliability	The architecture should define the model update engine only with processes that have proved to achieve the goal for which the architecture includes the processes.
CAR2	Enable smooth development, Uniform	The architecture should mark parallel processes that have to run at the same time.
CAR3	Enable smooth development, Uniform	The architecture should mark processes that require collaboration between a clinical end user and a CDSS.
CAR4	Enable smooth development, Uniform	The architecture should be designed according to one description language.
CAR5	Enable smooth development, Uniform	The architecture should distinguish components that produce information and components that consume information.
CAR6	Enable smooth development, Completeness	The architecture should inform how the architecture is used for development in specific healthcare decision-making contexts.
CAR7	Enable smooth development, Completeness	The architecture should inform on all processes that are needed to achieve the goal of the CDSS.
CAR8	Enable smooth development, Completeness	The architecture should inform on all data objects associated with the architecture.
CAR9	Enable smooth development, Completeness	The architecture should inform on all dependencies between architecture components.
CAR10	Modifiable	The architecture should be adaptable to all preferences present in healthcare decision-making contexts.
CAR11	Modifiable	The architecture should be adaptable to all potential future preferences in potential healthcare decision-making contexts.
CAR12	Compatible	The architecture should always be integrable with the service environment of Council.
CAR13	Compatible	The architecture should define a CDSS that is implementable within four weeks.
CAR14	Compatible	The architecture should ensure choices are not retrievable to an individual clinical end user and Council only has access to decision-making behaviour for which end users provided permission.

be able to try out an update and undo the outcomes of the update by resetting the active recommendation generator model. As a result, clinical end user’s understanding of the components and outcomes of a dynamic BAIT-based CDSS increases (Siau & Shen, 2003).

In the context of a dynamic BAIT-based CDSS, trust cannot be created at once. Trust rather develops over time (Siau & Wang, 2018). As such, the initial trust needs to be nurtured during ongoing use (Gefen et al., 2003; Siau & Wang, 2018). The development of trust heavily depends on a CDSS’s reliability, degree of collaboration, security, and transparency (Aoki, 2020; Siau & Shen, 2003). The previous paragraph covers the transparency of a CDSS. In the context of a dynamic BAIT-based CDSS, reliability implies that an update ensures the CDSS adjusts its information processing according to the contextual change as intended.

Collaboration requires a BAIT-based CDSS never to exclude partnership with clinical end users (Alexander, 2006; Siau & Wang, 2018). Collaboration is important because clinical end users are hesitant for a CDSS that ignores the end user’s presence. A CDSS that takes over control forms a threat to the user’s professional autonomy (Esmaeilzadeh, Sambasivan, Kumar, & Nezakati, 2015; Friedberg et al., 2014; Sambasivan, Esmaeilzadeh, Kumar, & Nezakati, 2012; Wang et al., 2021). The interviews show that this threat is not solely associated with tasks that need the knowledge and skill of a clinical end user. Instead, the tasks that need more generic skills, like updating maintenance tasks, also impact the end user’s perceived autonomy. Having control over these tasks allows clinical end users to know what goes on and to understand the CDSS better (see Appendix B). As such, the architecture of a dynamic BAIT-based should not only define independently working processes. Moreover, the end user should always be in the position to select the choice recommendation generator model that the CDSS operates.

Similarly, too much influence of Council will cause a comparable threat to the end user’s autonomy. Council mainly manages the static BAIT-based CDSS. For instance, Council manually assesses the model performance and discusses the outcome with a clinical end user. The enhancement of trust in the CDSS will benefit from a minimization of the intervening actions performed by Council that are beyond the control of the clinical end user. However, the minimization should

only concern actions of Council that the end user did not request explicitly. By doing so, the minimization of Council's input does not affect clinical end users' support requests (see section 3.2.3).

Finally, the security of the CDSS and the processed data plays a vital role in nurturing trust (Kusumasondjaja, Shanka, & Marchegiani, 2012; Siau & Shen, 2003). Because the current CDSS is built in an environment that deals with CDSS security, the security is considered valid by ensuring that the architecture guides implementation within this environment. The Client Artefact Requirements cover this compatibility with the existing environment (see section 3.2.3).

Ease of use

According to Venkatesh (2000), the user acceptance of an AI tool depends heavily on the ease of use of the tool. A CDSS that requires too much effort from clinical end users will conflict with the clinical end users' goals. The CDSS is not beneficial for either clinical end users or patients and becomes a disturbing factor rather than an assisting tool as soon as a CDSS unnecessarily occupies valuable time that clinical end users could use for patient care (Castillo & Kelemen, 2013; Yao & Kumar, 2013). The ease of use of a CDSS is inversely related to the complexity of the CDSS (Keil, Beranek, & Konsynski, 1995). Rogers and Shoemaker (1971) define the complexity of a CDSS as the degree to which the CDSS is perceived as relatively difficult to understand and the effort needed to use the CDSS properly. Having the understandability component covered (see section 3.2.3), ease of use in the context of a dynamic BAIT-based CDSS comes down to the complexity of the actions required to fulfill the clinical end users temporal goal and the time the end user needs to do so.

Supportability

The provision of continuous assistance in using the CDSS decreases the anxiety towards the CDSS and avoids that clinical end users will reject adopting the CDSS (Castillo & Kelemen, 2013; Anderson & Willson, 2008). Assistance decreases anxiety and frustration of end users and fosters a positive attitude towards the support CDSS (Castillo & Kelemen, 2013). Castillo and Kelemen (2013) even state that for clinical end users to accept and effectively use a CDSS, the presence of support is vital. Therefore, clinical end users must believe support is always available as soon as the end user does not understand the CDSS processes or outputs.

Service availability

The service availability refers to the CDSS being ready to perform the tasks it must execute to deliver the expected outcomes (F. Tabares et al., 2017). The availability covers the consultation of choice recommendations as well as any updating component. Regarding the first aspect, the architecture should avoid the situation where a clinical end user needs to choose between treatment options urgently and the BAIT-based CDSS is out of order. Concerning the second aspect, the architecture should ensure that an update can always be executed when desired.

Development Guiding Requirements

The DGRs drive the architecture design to incorporate certain CDSS functionalities and qualities. As such, these requirements further outline the design of the main components in the architecture design space framework (see chapter 2) in the context of a dynamic BAIT-based CDSS. The interview analysis identifies the DGRs that the architecture should meet to guide the development of each main CDSS component in the right way. Table 3.6 presents an overview of the DGRs per component at the end of this chapter.

Adaptive choice base

The adaptive choice base enables the CDSS to store experiment choices and a continuous inflow of real-life choices. While doing so, the choice base should separate experiment choices from real-life choices because both choice types are produced and consumed by different kinds of processes. To avoid that the architecture incorporates processes that consume the wrong choice type, all processes in the architecture should be able to recognize choice types directly. Moreover, choices differ in the additional features they possess. For instance, the date on which a clinical end user made the choice. Therefore, the choice base should be capable of storing the features attached to each choice.

As chapter 2 describes, real-life choices may be incomplete. Clinical end users did not give a method for dealing with missing data. However, the interviewed end users did make explicit that data cannot be interchanged by filling in missing values with averages calculated over other patients in the choice base. According to Washington, Ravulaparthi, Rose, Hensher, and Pendyala (2014), one of the approaches to obtain information on missing attribute values is to find solutions

Table 3.5: *Quality Attribute Requirements*

Reference	Rationale	Requirement
QAR1	Trustability	The architecture should always contain processes that work in partnership with clinical end users.
QAR2	Trustability	The architecture should minimize the number of intervening actions needed from Council that are not requested by a clinical end user.
QAR3	Trustability	The architecture should never force a fully automated updating component.
QAR4	Trustability	The architecture should avoid the development of a CDSS that forces clinical end users to accept a choice recommendation generator model version.
QAR5	Trustability	The architecture should only include components and provide clinical end users with outcomes that a clinical end user without any knowledge about Discrete Choice Modelling, statistics, and Machine Learning can understand.
QAR6	Trustability	The architecture should force the development of a CDSS that makes its internal changes transparent for clinical end users.
QAR7	Trustability	The architecture should avoid the development of a CDSS that allows clinical end users to directly determine the importance of a single real-life choice in the model estimation.
QAR8	Trustability	The architecture should force the development of a CDSS that only updates the choice recommendation generator model according to contextual changes captured by the experiment choices and real-life choices the CDSS is informed about.
QAR9	Trustability	The architecture should only define components that work statistically correct.
QAR10	Ease of use	The architecture should minimize the time and number of activities that a CDSS requires from clinical end users to fulfil a clinical end user's goals with the CDSS.
QAR11	Ease of use	The architecture should force the development of a CDSS that only asks a clinical end user to enter choice-specific data when entering a real-life choice from which the clinical end user will benefit later in time.
QAR12	Service availability	The architecture should force the development of a CDSS that clinical end users can always use for a choice recommendation request and measurement of a real-life choice.
QAR13	Supportability	The architecture should force the development of a CDSS that enables clinical end users to always request online support on the CDSS components and outcomes these components produce.

to impute the missing values with the average of the observed values. However, because medical patient data is specific to a particular patient, information that stems from other patients cannot replace the missing values. Moreover, if the flow of incoming real-life choices is marginal, the amount of available information is too small for a valid average calculation (Washington et al., 2014). As an alternative to imputing missing values with the average of observed values, Steinberg and Scott Cardell (1992) show that it is possible to estimate binary discrete choice models by pooling the accessible real-life choices with commonly available public-use data. However, this gives rise to the same problem: the missing values are patient-specific. In sum, the CDSS should deal with missing values that other data sources cannot cover. As a result, the CDSS must cope with missing values for choice attributes during the estimation process instead of modifying the choice data with imputation solutions.

Model update engine

The model update engine should complete two main tasks. First, the goal of the update engine is to ensure that the recommendation of a dynamic BAIT-based CDSS remains relevant in a changing context. Therefore, the update engine should estimate new parameters as soon as a clinical end user believes the CDSS needs an update. The interview outcomes show that the opinions about the extent to which particular choice types should influence the update vary among contexts. Clinical end users want to determine the influence that specific types of choices have on the model estimation and exclude particular choices from the update. Therefore, the architecture should specify a model estimation process that can incorporate a weighted selection of choices rather than all choices stored in the adaptive choice base.

Second, the engine should take care of the performance validation of the updated model. The interview analysis shows that clinical end users and Council both want to track the performance of successive model updates. In addition, the interview analysis indicates that the architecture should specify distinct metrics to communicate the performance validation outcomes to Council

and the clinical end users. This need for different treatments follows from the stakeholders' deviating interests. Clinical end users are interested in the extent to which the CDSS generates suitable choice recommendations in the present context. As such, clinical end users want to know how often a choice recommendation aligns with the clinical end user's choice. On top of this insight, Council also pursues insight into the model diagnostics representing the choice model's goodness of fit with the choice data. By doing so, Council can track the goodness of the estimated models resulting from updates in different contexts over time and gains experience in how a dynamic BAIT-based CDSS behaves in different decision-making contexts. Therefore, Council should have insight into both the performance metrics and the DCM metrics that reveal the goodness of fit.

The interviews indicated that a valuable validation of a dynamic CDSS only includes accurate choices. The inclusion of accurate choices ensures that the model validation process assesses the performance of the recommendation generator model for the present decision-making context. As a result, the validation outcomes will indicate whether the CDSS is flexible enough to capture changing perspectives. If the validation process would incorporate all available choices that the CDSS stores, the validation outcome only reports whether the enhanced amount of choice information enabled the CDSS to understand the decision-making context better. Moreover, including all choices makes a temporal performance decrease likely: it is theoretically impossible to incorporate the renewed contextual knowledge so quickly that the CDSS will never make a few more mistakes than it did before. As a result, this temporal decrease may give a biased representation. Validating the model only with recent choices avoids this seemingly temporal performance decrease because the update informed the model with new choice information. As such, it should be better aligned with the present choice tasks than the previous model was. The validation on recent real-life choices has two implications:

1. The choices that clinical end users entered to the choice base latest are the most recent choices available for the validation. Therefore, a valuable validation process should include these choices. However, including the choices that clinical end users entered most recently does not guarantee the inclusion of choices with a particular age. If a choice task occurs rarely, the inflow of new real-life choices is low. The inclusion of the latest choices might imply a validation set of choices from, for example, half a year ago. Therefore, clinical end users should have insight into the date at which a clinical end user entered a choice used for the validation. By doing so, end users can judge the value of the validation.
2. Experiment choices are collected at a particular point in time. Depending on the degree of dynamism in the context, they will lose their accuracy. Moreover, they originate from a controlled experiment. For these two reasons, they do not represent fair samples to validate the recommendation model performance in real life.

Finally, the choices used for the validation must be new to model (Ibrahim & Bennett, 2014; Rajer-Kanduč, Zupan, & Majcen, 2003; Raschka, 2018; Xu & Goodacre, 2018). Mixing estimation and validation choices would typically introduce an optimistic bias due to overfitting: it becomes ambiguous whether the model memorizes the estimation choices or whether it generalizes to new unseen real-life choices as aimed for (Raschka, 2018). To conclude, the validation should include the most recent real-life choices that were not part of the model estimation.

Model quality monitor

The architecture should specify a quality monitor to assess the performance of the BAIT-based CDSS constantly. In the context of a dynamic BAIT-based CDSS, such a monitor assesses if the generated recommendations align with the clinical end users' choices. As chapter 2 explains, for each choice a clinical end user enters to the CDSS the monitor should compare the CDSS's choice recommendation with the threshold and the clinical end user's answer to that choice. The interview analysis identified three considerations that further shape the design of the model quality monitor:

1. The majority threshold. This threshold determines if the end user will perceive the CDSS's recommendation as a vote in favour or against the treatment. The interviewed clinical end users disagreed on the percentage representing the majority view. Therefore, clinical end users seem to have varying preferences regarding the threshold. Moreover, another threshold may be preferred when the end user wants the CDSS to be stricter on performance later in time. Therefore, the threshold should also be variable over context and time.
2. The level of acceptance. This level represents the maximal number of deviating recommendations a CDSS may generate until the end user believes an update is needed. The interview

analysis shows that clinical end users are hesitant to name a value for the level of acceptance because they have no experience with the performance of a dynamic BAIT-based CDSS in their context. Consequently, the level of acceptance should also be variable over context and time. Two aspects further substantiate this. First, it differs per context how quickly a dynamic BAIT-based CDSS adapts to new contextual knowledge. The more time the CDSS needs to adapt, the sooner it will reach the level of acceptance. Second, the inflow of real-life choices may be diverse over contexts. In some healthcare contexts, clinical end users encounter a choice task every hour, while other choice tasks only occur once a month. The higher the inflow, the sooner a CDSS will reach a particular level of acceptance. Given these two aspects, a clinical end user might prefer the level of acceptance to be stricter or wider.

3. The establishment of measures that should be taken in case the monitor does particular findings. The quality monitor has two kinds of findings. First, the monitor highlights all real-life choices for which the CDSS's recommendation did not correspond with the clinical end user's choice. These will be referred to as deviating recommendations in the remainder of this report. Second, the monitor notices if the level of acceptance is reached. The interview analysis shows that all interviewed clinical end users are interested in further investigating the real-life choices for which the CDSS's recommendation did not correspond with the clinical end user's choice. This investigation allows clinical end users to observe the choices for which the CDSS could not generate a corresponding recommendation or for which the clinical end users performed remarkably. To realize this investigation, the monitor should store each real-life choices for which the recommendation of the CDSS deviated from the choice made by the clinical end user. The consequences of a noticed performance decline are related to Human-Computer Interaction (HCI). The section below presents the requirements concerning HCI.

Human-Computer Interaction

The HCI component covers the interaction between a dynamic BAIT-based CDSS and a clinical end user. The interview analysis indicates that three types of information are relevant for clinical end users: insight into physicians' decision-making behavior, internal model changes, updating activity, and model performance. Regarding the first, clinical end users are interested in reflecting on how decision-making within the clinical team evolves. Therefore, clinical end users need to be able always to access all model estimations ever stored. Moreover, the differences in decision-making behaviour between subgroups is of interest—for instance, insight into the differences between senior and junior physicians. Concerning the second, clinical end users want the CDSS's changes to be tractable as section 3.2.3 captures. With regard to the third, clinical end users like to be informed about a completed update to be sure that the CDSS operates an accurate model. With regard to the fourth, clinical end users want to know if the model quality monitor notices that the performance has declined below the level of acceptance. Because clinical end users are occupied during working hours, alerts that need action should remain visual until the associated action has been successfully performed (Khalifa, 2014).

The information that the CDSS provides to the clinical end users should always be brief. Notifications with a high information density will cause an alert fatigue (Castillo & Kelemen, 2013). Moreover, the provision of dense information that approaches clinical end users without being actively requested can slow down the workflow, efficiency and quality of the clinical end user (Castillo & Kelemen, 2013). Therefore, the CDSS should not inform clinical end users about details on the reason for the notification. Moreover, the CDSS architecture should limit the CDSS's communication of information that an end user did not actively request confirmation or plain alert. Finally, unrequested information should only concern a performance decline or updating activity.

Due to the variety of healthcare contexts (Eapen, 2021), the preferred level of updating automation varies over contexts. The interview analysis shows that all interviewed end users want to receive an alert from the CDSS if the CDSS reaches the level of acceptance. However, some clinical end users prefer that the updating process is automatically triggered as soon as the performance has declined below the level of acceptance, where others get anxious about a self-updating CDSS. Therefore, the activation of the updating process must be implementable according to various levels of automation. QAR4 covers the flexibility of the CDSS's level of updating automation (see section 3.2.3).

3.3 Summary chapter 3

The goal of chapter 3 is to identify and formulate the requirements of a dynamic BAIT-based CDSS architecture for Council. To this end, this chapter first explicates why Council deems the static BAIT-based CDSS Council currently offers to clinical end users insufficient. The findings show that the static BAIT-based CDSS lacks the following features that make suitable for dynamic healthcare contexts:

- The BAIT-based CDSS does not retain its accuracy over time, because there is no established updating mechanism that lets a choice recommendation generator model incorporate developments in the decision-making context.
- The BAIT-based CDSS cannot estimate a choice recommendation generator model based on a combination of experiment choices and real-life choices, which are both collected under different conditions.
- The BAIT-based CDSS cannot deal with the characteristics of choices collected in a real-life setting. For instance, real-life choices may contain missing values.
- The BAIT-based CDSS does not assess the performance of a choice recommendation generator model in a changing context on a continuous basis.

Council perceives covering these lacking features as challenging because of the novelty of DCM in the context of decision support, the variety of healthcare contexts, and the expected change in the preferences of clinical end users applying a BAIT-based CDSS over time. To tackle these challenges, Council needs a dynamic BAIT-based CDSS architecture that describes how Council can develop CDSSs that fit the dynamic healthcare contexts and adhere to the changing preferences present in the varying healthcare contexts. With this guiding tool, Council has the groundwork for developing a dynamic BAIT-based CDSS for each potential healthcare client.

The architecture design space framework presented in chapter 2 provides an initial insight into the main components of a dynamic BAIT-based CDSS architecture. To further specify the architecture as Council desires it, the research continued with six interviews and a literature review. The participants were the key stakeholders: representatives of Council and clinical end users. The interviews and the literature review resulted in an initial list of architecture requirements. This report distinguishes three types of architecture requirements: Client Artefact Requirements (CAR), Quality Attribute Requirements (QAR), and Development Guiding Requirements (DGR). Along with the design process, lessons learned refined the initial list of requirements. The final list of architecture requirement provides an answer to the third sub-question:

3. What are the requirements for a CDSS architecture of a dynamic BAIT-based CDSS?

The set of architecture requirements functions as the final design outline of the main components that the architecture design space framework in chapter 2 presents. As such, the architecture requirements define the architecture solution for Council that chapter 5 presents. The realization of the architecture requirements demanded a series of design decisions. Chapter 4 presents these design decisions.

Table 3.6: *Development Guiding Requirements*

Reference	Rationale	Requirement
DGR1	Adaptive choice base	The architecture should force the development of a CDSS that distinguishes experiment choices and real-life choices.
DGR2	Adaptive choice base	The architecture should force the development of a CDSS that does not interchange patient-specific data to deal with incomplete real-life choices.
DGR3	Adaptive choice base	The architecture should force the development of a CDSS that stores a choice with all features assigned to the choice when a clinical end user entered the choice into the CDSS.
DGR4	Model quality monitor	The architecture should force the development of a CDSS that compares each choice recommendation with the majority threshold and the clinical end user's choice as soon as an end user enters a real-life choice into the CDSS.
DGR5	Model quality monitor	The architecture should force the development of a CDSS that copies real-life choices exceeding the majority threshold but deviate from a clinical end user's choice to a separate database.
DGR6	Model quality monitor	The architecture should force the development of a CDSS that operates a modifiable majority threshold and level of acceptance.
DGR7	Model update engine	The architecture should force the development of a CDSS that estimates a new choice recommendation generator model according to the choice types and weight specification clinical end users selected as soon as clinical end users deem this model inaccurate or undesired for decision support.
DGR8	Model update engine	The architecture should force the development of a CDSS that assesses the performance of a newly estimated choice recommendation generator model based on a unique set of recent real-life choices.
DGR9	Model update engine	The architecture should force the development of a CDSS that makes the date at which a clinical end user entered a choice used for the model validation transparent.
DGR10	Model update engine	The architecture should force the development of a CDSS that allows replacing the experiment choices with experiment choices from a new choice experiment.
DGR11	Model update engine	The architecture should force the development of a CDSS that enables clinical end users to request the performance metrics for all choice recommendation generator model updates.
DGR12	Model update engine	The architecture should force the development of a CDSS that gives Council insight into the performance metrics for the choice recommendation generator model updates of all healthcare contexts.
DGR13	HCI component	The architecture should force the development of a CDSS that presents an alert to clinical end users when the level of acceptance has been reached and the choice recommendation generator model is not updated yet.
DGR14	HCI component	The architecture should force the development of a CDSS that confirms the completion of a choice recommendation generator model update.
DGR15	HCI component	The architecture should force the development of a CDSS that allows clinical end users to request the relative importance of the choice attributes of all choice recommendation generator model updates.
DGR16	HCI component	The architecture should force the development of a CDSS that allows clinical end users to request the relative importance of the choice attributes for a by the clinical end user selected subgroups.

Chapter 4

Specification of the Architecture: Design Decisions

This chapter presents the structural specification of the architecture design. For some of the architecture requirements, the solutions are trivial. However, other requirements gave rise to contradictions or solutions that need additional argumentation. This section focuses on the nontrivial decisions made to adhere to all architecture requirements because these decisions are not straightforward. First, section 4.1 presents the nontrivial design decisions related to the structure of the architecture. Next, section 4.2 lists the decisions on the estimation and validation of new model updates. Third, section 4.3 informs on the design decisions regarding the customization of the update engine. Finally, section 4.4 presents the decisions concerning the management of all information with which a dynamic BAIT-based Clinical Decision Support System (CDSS) deals. The chapter concludes with a summary of all design decisions in section 4.5. The specification of the architecture functions as a first step towards answering the fourth sub-question:

4. What does a system architecture of a dynamic BAIT-based CDSS look like?

4.1 Design decisions on the architecture structure

This section presents the design decisions that impact the structure of the architecture. First, subsection 4.1.1 explains the layered design of the architecture. Next, subsection 4.1.2 explains why the architecture design follows an adaptable approach. Finally, subsection 4.1.3 argues why the architecture does not specify any particular software and hardware prerequisites.

4.1.1 Layered design

Decision. The architecture has a layered structure. Moreover, this layered structure is defined in terms of the Architecture Description Language (ADL) ArchiMate.

Argument. An Architecture Description Language (ADL) is a language that describes the software and hardware architecture of a system. The description may cover software features such as processes, data, and subprograms and hardware components such as processors and devices (Björnander, 2011). Examples of ADL's are ArchiMate, AADL, ACME, Rapide, Darwin, Aesop, TASM, or UML. Compared to the other ADL's, ArchiMate encourages flexibility as it allows future additions or adaptation to different application contexts with different extensions. ArchiMate is a flexible ADL because it aligns with the concept of a service-oriented architecture (SOA). SOA supports flexibility by combining and reusing existing services to adhere to demands that change over time (Meertens, Iacob, & Nieuwenhuis, 2010). ArchiMate incorporates SOA by distinguishing three layers at which "loosely coupled" components are specified: business, application, and technology layer (Arsanjani, 2004; Lu, 2005). Each layer in the layered architecture pattern has a specific role and responsibility within the application.

1. The business layer: represents the business processes stakeholders perform to interact with a system or complete tasks related to a system.

2. The application layer: represents all software processes that support the business layer with application services that software applications realize.
3. The Technology Layer: represents infrastructure services needed to run applications at the application layer, like computer and communication hardware and system software.

The “loosely coupled” components can be changed or replaced without affecting other parts of the architecture (Pombo Jimenez, 2017). As such, it allows for the adaptability of the architecture in changing conditions and over time. Moreover, ArchiMate is a visual architecture description language, which is helpful when communicating the architecture to Councilyl. More specifically, it allows for an ordered structure with the definition of components per layer to explicate parallel processes and processes in which humans are involved. By doing so, it enhances the understandability for Councilyl. Finally, ArchiMate is a widely used language to describe the construction and operation of business processes, organizational structures, information flows, IT systems, and technical infrastructure. Especially in the context of CDSS design (Power, 2002).

4.1.2 Adaptable design

Decision. The architecture has an adaptable structure.

Argument. The architecture must guide the development of different CDSS’s that satisfy the preferences in different contexts. A layered approach already allows removing or replacing components in the architecture (see subsection 4.1.1). However, the layered approach does not give the architecture multiple variants of CDSS features to satisfy varying needs. On the contrary, an adaptable structure supports designing optional extensions that supplement the essential architecture components. Because these extensions are optional, an adaptable architecture allows CDSS developers to answer the varying preferences in different healthcare contexts. Moreover, an adaptable architecture allows Councilyl to dynamically include additional CDSS components when these components turn out to be preferred by end user over time (Madura, 2006). As a result, all features that the core of the architecture covers are static, meaning they are fixed and independent of changing needs over time. On the contrary, the extensions in the architecture are dynamic since the implementation depends on the end user’s preferences which may change over time.

The adaptable structure also matches the research process in three ways. First, it allows the designer to start with a limited design scope while leaving space for the satisfaction of additional preferences concerning the dynamic support as being revealed over time (Moore & Chang, 1980). Second, this approach suits the ADR research method. Both the adaptable approach and ADR accept that the design specification is not well-defined in the beginning (Moore & Chang, 1980; Sein et al., 2011). Instead, the ADR methodology presumes that the initial requirements identification is geared to provide sufficient information to build only the nucleus of the architecture (Sein et al., 2011). Refinements are identified during the design process designer (Sein et al., 2011). Finally, an adaptable architecture design allows Councilyl only to implement the specific parts of the architecture needed to satisfy the end user’s preferences. With an architecture that requires a complete implementation to work properly, time will be wasted on implementing functionalities that do not match the end user’s necessities. As a result, an adaptable architecture saves a CDSS provider’s time.

4.1.3 Soft- and hardware independence

Decision. The architecture does not specify software and hardware prerequisites at a technology layer.

Argument. The architecture needs to be adaptable over time. Software technologies and hardware technologies develop quickly, and many novel innovations arise over time. For instance, Councilyl may switch from the software platform used to code and host the decision support service or use another processor. Another advantage of the components’ independence from underlying technology is that Councilyl’s developers can build and change the components in the way they find most cost-effective and timely. Therefore, architecture design does not specify components, like a software development platform or any hardware components, at a technology layer (for an explanation of the different layers, see subsection 4.1.1).

4.2 Design decisions on the estimation and validation of recommendation generator model updates

4.2.1 Extensions involvement and update trigger automation

Decision. The architecture defines three extensions that allow for a more and less automated activation of the model update engine.

Argument. The architecture should not force an automated activation of updates. Four alternative options to activate an update exist. The option that should always be possible forms the default: a manual activation by an end user without a trigger from the CDSS. Despite the desired level of updating automation, an end user should always be able to update the CDSS. As a result, the architecture includes three model update engine activation extensions:

1. Activation Extension 1: Manual update activation triggered by an update request provided by the CDSS. The CDSS triggers this activation as soon as it reaches the level of acceptance. The end user can activate an update by confirming the request.
2. Activation Extension 2: Automated update activation as soon as the CDSS reaches the level of acceptance.
3. Activation Extension 3: Periodic automated update activation.

Activation Extension 1 and 2 require a trigger from the model quality monitor that checks if the CDSS has reached the level of acceptance. Extension 3 requires a user setting specification concerning the frequency with which the end user wants the CDSS to update. The frequency can either be defined in terms of time or in terms of the number of new real-life choices added since the previous update. By doing so, the architecture allows the development of a CDSS that matches the degree of change in the context. If the context is highly dynamic, the CDSS can even activate updates on a real-time basis.

A CDSS can combine Activation Extension 2 with Activation Extension 3. By doing so, the architecture avoids the situation in which the model update engine will never update because the CDSS never reaches the level of acceptance. When a CDSS includes both Activation Extension 2 and Activation Extension 3, the CDSS will activate an when it reaches the level of acceptance and on a to be specified periodic basis as a backup that guarantees the activation of updates.

4.2.2 Pooled estimation

Decision. The architecture specifies a pooled approach for the model estimation.

Argument. Two main techniques for a model estimation with experiment choices and real-life choices exist: “pooled” and “sequential” estimation (J. J. Louviere, Hensher, & Swait, 2000). The techniques differ in how they combine the estimated parameters estimated with either the experiment choices or the real-life choices (Axsen et al., 2009). The pooling approach combines both sources to estimate the parameters from both sources at the same time. With the sequential approach, on the contrary, separate experiment and real-life choice models are estimated (Axsen et al., 2009; Swait et al., 1994). The parameters estimated with the experiment choices form the basis. Only the constant representing the utility that the parameters did not capture stems from the real-life choice based model. The pooling approach better suits the goal of the architecture for three reasons. First, the architecture aims to inform the parameters by both experiment and real-life choices, which is only possible with the pooling approach (Axsen et al., 2009). Second, the pooling approach allows weighing the experiment choices and real-life choices differently during the model estimation. This difference in weight is needed to address the varying preferences among healthcare contexts (see section 4.3). Finally, the sequential process is more complex. This complexity conflicts with the aim for a transparent and understandable CDSS (see requirement QAR5 in section 3.2.3). Incorporating both techniques would, for the same reason, result in a too complex service. Primarily because in the case of two alternatives, the clinical end users will need to make a choice between the two statistical options, which is out of their comfort zone.

4.2.3 Restricting the individual parameters with missing values to zero

Decision. The architecture deals with incomplete data during the estimation process by restricting the individual parameters of missing attributes to zero. By doing so, only attributes for which

values are known can shape the model estimation.

Argument. The decision stems from two reasons. First, there are no adequate alternatives. An option to deal with incomplete real-life choices is to exclude all real-life choices with unknown attribute values. However, the inflow of real-life choices may be scarce in particular healthcare decision-making contexts. Second, restricting the parameters to zero makes the model estimation a more objective representation of reality. When an end user enters an incomplete real-life choice, the clinical end user was able to choose even though not all information about the patient was available. Therefore, a missing value for a choice attribute indicates that the attribute was not important in the choice or the choice was easy because of the (extreme) scores on other choice attributes (Rizzi & de Dios Ortúzar, 2003; Rosenberger, Peterson, Clarke, & Brown, 2003). Despite the root cause, a missing value for an attribute indicates that the attribute did not influence the physician’s choice (Carlsson, Kataria, & Lampi, 2010). Accordingly, it would be realistic if a dynamic BAIT-based CDSS also deals with the attribute as it was not relevant for the specific choice task. Restricting individual parameters for the missing attributes to zero allows excluding these attributes as influencing factors ???. The outcome resulting from a model estimation is then only a function of the attributes that the end user considered to make a choice in real life (DeShazo & Fermo, 2004; Campbell, Hutchinson, & Scarpa, 2006).

A note regarding the restriction of parameters should be necessary. One may argue that the restriction of an individual parameter for attributes end users ignored to zero is too restrictive. In previous work, researchers asked respondents to define whether they ignored specific attributes in their choice. It turns out they often mean that they have just put less weight on the attribute they claimed to have ignored (Carlsson et al., 2010). However, in the context of a dynamic BAIT-based CDSS, it is known that the attribute was not considered by the clinical end user, because it was not known. The restriction of the parameter for that attribute to zero forms an admissible representation.

4.2.4 Selection of performance metrics

Decision. The architecture provides clinical end users with insight into the Correspondence rate and the Agreement table and Council with insight in all anonymous performance metrics defined in section 2.2.2.

Argument. Both Council and clinical end users want insight into the performance of an updated recommendation generator model. Section 2.2.2 presents a set of proved performance metrics. However, some of these metrics require knowledge and skill in statistics, Discrete Choice Modeling (DCM), or Machine Learning (ML). However, the architecture should only provide end users with CDSS outcomes that they can understand without knowledge and skill in these areas (see requirement QAR5 in section 3.2.3). Therefore, the architecture only allows end users to access the Correspondence rate (accuracy) and the Agreement table (see section 2.2.2). These metrics are both easy to comprehend without any statistical, DCM, or ML knowledge. Because end users will not be familiar with the terms Correspondence rate and Agreement table, the architecture recommends that CDSS developers provide the metrics presentation with an explanation in the interface. The architecture ensures that Council has insight into all metrics.

4.2.5 K-fold cross validation with manipulated data split

Decision. The architecture describes a manipulated k-fold validation process for the performance assessment of each model update.

Argument. The decision stems from three main reasons. First, the manipulated k-fold manipulated k-fold validation avoids a biased performance assessment. The k-fold validation process is a ML technique for the validation of a model trained with a particular data set (Raschka, 2018). The k-fold validation process partitions the complete data set into k folds to then iterate over the data set k times (Raschka, 2018). Each iteration, the process divides the data set into k parts. One part is held back as the validation set. The remaining k-1 parts are used for the model training. After the k iterations, the results of the k runs are averaged to produce a single validation estimation (Grimm, Mazza, & Davoudzadeh, 2017; Jung, 2018). Common practice is to use 1/3 of the sample size for the model training (Wong, 2015). The k-fold validation ensures that the validation includes the complete data set. Using the complete set significantly reduces the bias and variance of the validation outcome (Abu-Mostafa, Magdon-Ismail, & Lin, 2012; Xiong et al., 2020). Moreover, the

confidence with which the CDSS can share the performance assessment with end users and Council increases. If the metrics stem from a single model validation, the result could be coincidence, stem from a biased validation set, or be the result of a manifestation of randomness in the choice set (Abu-Mostafa et al., 2012).

Second, the k-fold validation technique is proven and popular for assessing the model performance in case of a small data set (Bengio & Grandvalet, 2004; Ibrahim & Bennett, 2014; Jung, 2018; C. Lee, Ran, Yang, & Loh, 2010; Wong, 2015). The k-fold process is even the most plausible candidate method in cases where a part of data cannot be withheld for validation (Sidiropoulos et al., 2012). Because some medical choice tasks occur rarely, the number of real-life choices a dynamic BAIT-based CDSS can use for an update might be limited. By incorporating the k-fold validation technique, the architecture ensures that a CDSS can always validate a model properly, even though the inflow of real-life choices is low.

Finally, alternatives to the k-fold are insufficient. The so-called hold-out technique splits the data set only ones: one set is used to train a model, and one set to validate that model (Bengio & Grandvalet, 2004; Xiong et al., 2020). By doing so, the technique does not allow the model training to include the complete data set. As a result, this technique uses training data inefficiently. Therefore, it should not be applied in contexts where the data set is small (Bengio & Grandvalet, 2004; Ibrahim & Bennett, 2014; Jung, 2018; C. Lee et al., 2010; Wong, 2015). A second disadvantage of the hold-out technique is that the hold-out technique might over-represent a particular real-life choice in either the training or validation set. For instance, a training set with only female patients. Accordingly, Xiong et al. (2020) state that the model performance can be biased due to the splitting if the data set is small as a consequence. In k-fold validation, each real-life choice has the opportunity of being tested (Raschka, 2018).

The k-fold validation is associated with two disadvantages in the context of a dynamic BAIT-based CDSS. First, the k-fold validation starts from the full choice set and splits this into k folds (Jung, 2018; Raschka, 2018; Xiong et al., 2020). However, the validation must be based on recent choices rather than on the complete choice base (see requirement DGR8 in section 3.2.3). A solution is to tune the k-fold validation with a manipulated data split. Instead of randomly splitting the complete set as if each choice in the choice contains the same amount of information, the validation process should control the selection of choices. For each iteration, the CDSS randomly picks 1/3 of the real-life choices that end users have entered into the CDSS since the previous model update (Wong, 2015). By doing so, the CDSS generates a validation set for a particular iteration. Then, the CDSS picks the remaining real-life choices in the choice base and joins these with the experiment choices. By doing so, the CDSS generates a training set for a particular iteration. If there are not enough new real-life choices available, the CDSS will pick the real-life choices end users entered since the second-last update, and so on.

Second, randomly picking real-life choices from the choice base gives rise to the risk of an imbalanced data set, especially in contexts where the inflow of real-life choices is low (Raschka, 2018). By randomly pick real-life choices, the training set might mainly contain choices for which end users chose to operate. This problem becomes worse if the set of real-life choices has a high class imbalance upfront. This upfront imbalance is not unlikely in a healthcare decision-making context: the choice for particular treatments may be rare. An option to deal with an imbalanced data set is to divide the choice set in a stratified fashion (Raschka, 2018). However, the usefulness of this method to solve the problem of an imbalanced real-life choice set in the context of a BAIT-based CDSS should be further researched from a statistical point of view. Therefore, dealing with an imbalanced choice set is left out of the scope of the architecture design in this research.

4.3 Design decisions on the customization of the update engine

4.3.1 Module extensions for choice inclusion and weight specification

Decision. The architecture defines three extensions to customize the inclusion of particular choice types and the weight with which particular choice types influence a model update.

Argument. The preferences concerning the choice types influencing a model update vary between healthcare contexts. End users have varying preferences on two axes. On the one hand, an update can include different types of choices. To this end, the CDSS can distinguish experiment choices and real-life choices. Moreover, the CDSS can distinguish different types of real-life choices with

a set of features (see Table 4.1). For instance, a clinical end user may prefer to exclude all choices made by junior physicians or all choices for which the CDSS generated a recommendation it was not confident about.

Table 4.1: *The features of real-life choices.*

Feature	Unit to distinguish choice types
Choice maker	User ID
Recommendation of CDSS	Minimal percentage
Date	Date interval
Time	Time interval
Deviation between the CDSS's recommendation and clinical end user's choice	Binary (yes or no)
Expertise level clinical end user	Category expertise (junior, senior)
Discipline level clinical end user	Category expertise (Context-dependent)
Confidence of clinical end user	Category (not sure, in between, super sure)
Confidence of CDSS	Binary (unsure= recommendation between 40% and 60%)
Choice made based on external factors	Binary (yes or no)

On the other hand, three alternatives for the weighting of different choice types exist. The weight of a choice type determines how much each choice of that type will influence the update. The first alternative is that end users can assign all choices with an equal weight. However, end users may prefer to specify the influence of particular choice types on an update. Therefore, a second alternative is to allow end user to specify the weight of particular choice types with a so-called importance rating. With this importance rating, end users specify how many times more a particular choice type should influence the model update than the choices of another type. For example, when experiment choices should influence the update twice as much as the real-life choices. A third alternative is to allow end user to specify the weight of particular choice types with a so-called importance balance. With this importance balance, an end users can balance choice types by specifying the shares of particular choice types in an update. For instance, when the update should be based on experiment choices for 60% and on real-life choices for 40%.

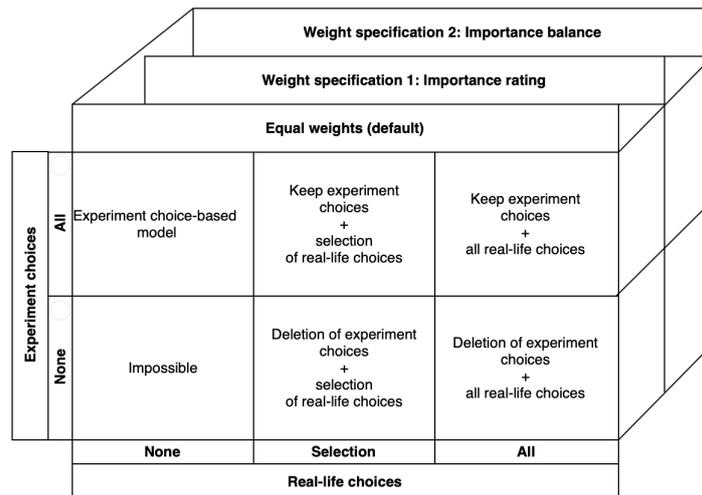


Figure 4.1: Conceptual overview of possible choice influence and weighting extensions.

The cube in Figure 4.1 visualizes the resulting options from combining different possibilities over the two axes mentioned above. Experiment choices can be either included or not. The same holds for real-life choices. Given the goal of this research, it is assumed that an update will always contain real-life choices. However, an update does not have to include all real-life choices: end

Weight specification	Real-life choices			
	All real-life choices	Selection of real-life choices		
		Changing the influence of a selection of real-life choice relative to the experiment choices	Changing the influence of a selection of real-life choices relative to the other real-life choices	Excluding the influence of a selection of real-life choices
Equal weight	Each experiment choice influences the model update with the same weight as each real-life choice	-	-	Each not-selected real-life choice does not influence the model update.
Weight specification 1: Importance rating	Each experiment choice influences the model update with another weight than real-life choices	Each experiment choice influences the model update with another weight than each choice in the selection of real-life choices	Each selected real-life choice influences the model update with another weight than each not-selected real-life choice	-
Weight specification 2: Importance balance	All experiment choices influence the model update with another share than real-life choices	The share of experiment choices in a model update is different than the share of the selection real-life choices	The share of selected real-life choices in a model update is different than the share of the not-selected real-life choices	-

Figure 4.2: Detailed overview of possible choice influence and weighting options.

users may prefer to exclude a subgroup of real-life choices with particular features. Given the choices that are included or excluded, the end user can assign a weight to choices of a particular type with one of the two weighting options. Given the goal of this research, it is assumed that an update will always contain real-life choices. Therefore, the focus is on the right two columns in Figure 4.1. Figure 4.3 gives a detailed explanation of the resulting options for the customization of a model update.

Legend	Weight specification	Real-life choices			
		All real-life choices	Selection of real-life choices		
			Changing the influence of a selection of real-life choice relative to the experiment choices	Changing the influence of a selection of real-life choices relative to the other real-life choices	Excluding the influence of a selection of real-life choices
Default information specification: no extension needed	Equal weight	Each experiment choice influences the model update with the same weight as each real-life choice	-	-	Each not-selected real-life choice does not influence the model update.
Information specification Extension 1	Weight specification 1: Importance rating	Each experiment choice influences the model update with another weight than (selected and not-selected) real-life choices	Each experiment choice influences the model update with another weight than each choice in the selection of real-life choices	Each selected real-life choice influences the model update with another weight than each not-selected real-life choice	-
Information specification Extension 2 Information specification Extension 3 Information specification Extension 1+3	Weight specification 2: Importance balance	All experiment choices influence the model update with another share than (selected and not-selected) real-life choices	The share of experiment choices in a model update is different than the share of the selection real-life choices	The share of selected real-life choices in a model update is different than the share of the not-selected real-life choices	-

Figure 4.3: Colored overview of possible choice influence and weighting options that marks extensions.

The realization of all update options in Figure 4.1 requires the definition of three extensions. Figure 4.2 visualizes the division of the extensions over the different options. Because these extensions allow end users to specify the choice information an update will include, they are referred to as Information specification Extensions. The list below gives an explanation of each extension. To support this explanation, Figure 4.4 gives a conceptual representation of each extension.

1. Information specification Extension 1: An experiment choice and real-life choice influence variation extension. An end user can specify a weight that adjusts the influence of the experiment choices respectively to the real-life choices. An end user can specify a weight in terms of the importance rating or the balance rating. As soon as the end user does not want experiment choices to influence the model update, the end user can use the importance balance to ensure the model update only incorporates real-life choices.
2. Information specification Extension 2: A real-life choice exclusion extension. This extension allows end users to specify a selection of real-life choices in terms of the choice features and

exclude the choices that do not have these features. This extension makes use of an AND operator, which means that the selection criteria add on. For instance, a clinical end user can request the exclusion of choices made by junior physicians and that were made between 10 pm and 5 am. Otherwise, the update will not realize a complete exclusion. For instance, the selection of choices would still include choices made by junior physicians because they were active and made choices 5 am and 10 pm.

3. Information specification Extension 3: A real-life choice and real-life choice influence variation extension. This extension allows end users to specify a selection of real-life choices in terms of the choice features and give this selection another weight than the real-life choices that do not have these features. The weight specification can be done in terms of the importance rating or the importance balance.

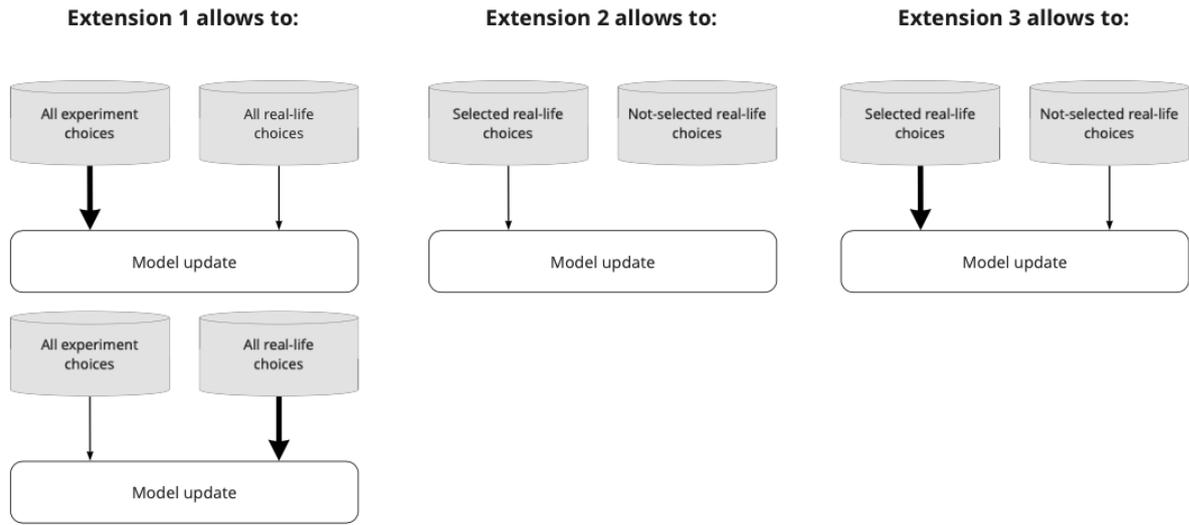


Figure 4.4: Conceptual overview of the three Information specification Extensions.

A CDSS can combine the implementation of Information specification Extension 1 with the implementation of Extension 2 and Extension 3. When combined with Extension 2, the selected real-life choices are weighted respectively to the experiment choices. When combined with Extension 3, the end user assigns a different weight to a selection of real-life choices. However, all selected and not-selected real-life choices are weighted differently compared to the experiment choices. By letting clinical end users choose extensions, clinical end users can shape the operation of a dynamic BAIT-based CDSS. As a result, a dynamic BAIT-based CDSS is receptive to the input of clinical end users.

4.3.2 Multiplication of sample size to realize the weight specification

Decision. The architecture describes a multiplication of the sample size to adjust the influence of choice types on a model update.

Argument. The customization of the influence of choice types with the extensions (see subsection 4.3.1) requires a mechanism to adjust the weights of these choice types in an update. The influence of a choice on an update depends on a set of factors. Factors are the number of choices, the correlation and variation among the observed choice attributes, the number of different choice sources, and the number of alternatives each choice set contains (Huber & Zwerina, 1996). Therefore, experiment choices will already have a different influence on an update than real-life choices without varying the weights of the choice types. The informative influence depends on the amount of econometric information that each choice sample contains. This econometric information differs per choice source because of the different conditions under which the collection of the choices took place. Generally, a set of experiment choices contains more information because they stem from an experimental setting where the goal is to design choice efficiently and with little bias as possible (Helveston et al., 2018; Cherchi & Hensher, 2015; Sanko, 2001). As a result, an enhanced number

of real-life choices will expand the influence of real-life choices on the model. However, this increase does not equal the increase in the influence of the addition of an experiment choice. Therefore, the balance of information between experiment and real-life choices does not completely depend on the sample sizes (Helveston et al., 2018).

However, the number of choices does determine the greater part of the informative influence on a model update. Therefore, it forms an effective mechanism to adjust the weights of the choice types in an update. In addition, varying the number of choices of a particular type of choice is understandable for end users without statistical knowledge. Finally, it is the only way possible to adjust the influence without creating biases. Two reasons further substantiate this. First, an option is to steer the influence via the number of choice alternatives of the experiment choices and real-life choices. However, this option is not feasible because the medical choice tasks have fixed choice alternatives. The choices are mainly about voting in favour or against medical treatment. Second, an option is to scale the model parameters estimated with the influence the end user specified. In this scenario, parameters are estimated based on different types of choices. For example, an update estimates a set of parameters based on experiment and a set of parameters based on real-life choices. Then, the CDSS pools and weights these parameters by the respective amounts of influence the end user specified. However, scaling the parameters with a weight specification leads to a biased model because the CDSS allows end users to manually alter the estimated model parameters.

Because of the reasons mentioned above, the architecture defines the influence of a choice type on the model type as the sample size of that choice type. Consequently, adjusting the influence of a choice type on a model update requires manipulating the number of times a choice of that type occurs in the set of choices included in an update. The architecture describes a CDSS that multiplies choices of a particular choice to vary their influence. The use of a choice multiplication stems from two reasons. The first reason is that an update cannot bisect choices. Therefore, the realization of a balance of choice types (for instance, 40% experiment choices and 60% real-life choices) cannot use half choices. Moreover, it is not recommendable to randomly delete choices from a choice set. The CDSS should not have the power to decide which choices will inform an update. Besides, randomly removing choices makes it unclear which information an update incorporates. Finally, the experiment choices are interdependent because they stem from a controlled environment (see subsection 1.2.2).

4.3.3 Temporary choice bases

Decision. The architecture defines a CDSS that copies choices for a model update from the original choice bases into temporary choice bases.

Argument. The architecture describes two temporary choice bases: a temporary experiment choice base and a temporary real-life choice base. For each update, the CDSS creates these temporary choice bases. The temporary choice bases contain all choices that will directly inform an update. Suppose the CDSS does not contain an Information specification Extension (see subsection 4.3.1). In that case, the temporary choice bases contain all experiment and real-life choices stored in the adaptive choice base. The CDSS does not multiply the choices in the temporary choice bases. However, when a CDSS includes an Information specification Extension, the temporary choice bases contain the choices in line with the information specification of an end user. The temporary choice bases allow a CDSS to collect the required choices for an update without deleting the original ratio of choice types. The reason for the design of temporary choice bases is that keeping the original choice bases available has advantages. The advantages are listed below:

- End users can always change from Information specification Extension because the original choice bases can be used to re-weight a selection of choice types.
- The k-fold validation process requires the original real-life choice bases, to avoid a biased performance assessment that includes double choice samples.
- End users can try out different choice selections and choice weights, because the original choice bases can be used to re-weight a selection of choice types.
- The temporary choice bases minimize the number of choices incorporated in the update. Every update, the CDSS creates new temporary choice bases by multiplying the original choice bases. By doing so, the number of choices is minimal compared to the situation in which the CDSS multiplies the same choice base over again for each update.

- The temporary choice bases enhance the availability of the CDSS. With temporary choice bases, the addition of new real-life choices to a CDSS’s adaptive choice is independent of the activity of the update engine. Because the update engine only uses the temporally choice bases, end users can add new real-life choices while the update engine is running.

4.3.4 Information specification Extensions as user settings

Decision. The architecture describes the Information specification Extensions as user settings the end user can access via an extension-specific interface.

Argument. By describing the Information specification Extensions as different interfaces, the end user can adjust the choice selection and choice weight specification without changing the implementation of the model update engine. When an end user applies a CDSS with Information specification Extension 1 (see subsection 4.3.1), the end user can adjust the influence of experiment choices respectively to real-life choices on an update via an interface. The CDSS stores the adjustment in the user settings. Each update, the CDSS checks the user settings. Accordingly, the CDSS creates the temporary choice bases by copying the correct choices from the original choice bases and multiplying these choices according to the weight specification in the user settings. Even switching between Information specification Extensions does not require changes in the structure of the model update engine. Council only needs to make a different interface needs available for the end user.

4.4 Design decisions on information management

4.4.1 The measurement of new real-life choices over time.

Decision. The architecture uses a modified version of the choice recommendation generator model that is part of the static BAIT-based CDSS for the measurement of real-life choices.

Argument. The existing static BAIT-based CDSS already contains a choice recommendation generator. This component requests clinical end users to specify the details of a choice task (see Figure 4.5). Therefore, this component is useful for the measurement of new real-life choices in the CDSS. Moreover, the component generates a choice recommendation based on the choice details entered by the clinical end user. To make the recommendation generator suitable for a dynamic BAIT-based CDSS, the architecture copies the recommendation generator of the static BAIT-based CDSS with two modifications.

First, the architecture expands the set of choice details that the recommendation generator of the static version requests from the clinical end user. The recommendation generator model asks clinical end user to enter the confidence with which he or she made a choice and if factors other than specified by the model influenced the choice (see Figure 4.6). By requesting this information, CDSSs can distinguish choices with additional features. Using these features, end users will be able to specify particular subgroups (subsection 4.3.1). For instance, an update that only includes choices about which end users were confident. Moreover, the confidence specification informs the calculation of the performance metric Confidence Representation.

Second, the architecture describes a CDSS that hides the recommendation with a “give me advice” button. This button allows clinical end users to enter a real-life choice without being influenced by the CDSS. The static CDSS automatically presents a recommendation to an end user when the end user enters the real-life choice details. However, the real-life choices an end users enter are likely to inform an update. End users might prefer to inform a CDSS update with real-life choices that the CDSS did not influence.

4.4.2 Separated choice bases for experiment and real-life choices

Decision. The adaptive choice base consists of two separate choices bases for experiment choices and real-life choices.

Argument. The argumentation for this decision stems from four reasons. First, the CDSS components and processes consume choices from different sources: experiment choices or real-life choices. For instance, the model estimation during a model update includes experiment choices and real-life choices. In contrast, the model validation during a model update consumes only real-life choices.

COUNCYL Decision Support

Test Klanc English (United Kingdom)

Choice advice Model management Notifications User settings Support

Add new choice
Enter the details on the choice task below and interpret the advice of Councyl based on these details afterwards.

Choice factors

- Age Unknown 30 ————— 70
- BMI Unknown <20
- Comorbidity Unknown Normal ————— Very limited
- Capacity Unknown Limited
- Frailty Unknown 1 and 2

Choice characteristics

Which choice did you make? NO YES

CHOICE ADVICE: 79%

Figure 4.5: The interface for a clinical end user to enter choice details and request a choice recommendation in the static BAIT-based CDSS.

Therefore, different processes need access to choices from different sources. Second, the construction of the temporary choice bases (see subsection 4.3.3) requires the separation of experiment choices and real-life choice choices. The temporary choice base might have to weigh the choices from one of the sources differently than choices from the other source. Finally, only an insufficient alternative exists. An alternative is that the process goes through the whole choice base and checks for each choice, whether it is an experiment choice or a real-life choice. By doing so, the run time of the update engine would increase. An increased run time is problematic in highly dynamic contexts where the contextual conditions change fast, or the CDSS has to update on a real-time basis.

4.4.3 Data accessibility for clinical end users

Decision. The architecture keeps the choices of a clinical end user hidden for clinical end users accessing a dynamic BAIT-based CDSS with a different user ID.

Argument. Information cannot be retrievable to an individual clinical end user. Therefore, end users should not be able to access the choices made by other individual clinical end users. The CDSS stores the choices of clinical end users with the feature user ID (see Table 4.1). Therefore, architecture should describe an additional measure to ensure that not all choices stored in the choice base are accessible by every clinical end user. To this end, the architecture describes an authorization check. As soon as an end user requests access to choices, this authorization checking process verifies the user ID of the end user requesting insight into the choices. Accordingly, the CDSS only makes the choices in the choice base visible that correspond with the user ID. An alternative is to remove the user account associated with the stored choices. However, this gives rise to the risk that an end user can still link a choice to a particular individual. For instance, because only one clinical end user was active when the end user who made the choice entered the choice to the CDSS.

4.4.4 Split in user settings Councyl and clinical end users

Decision. The architecture describes a data object representing the number of available subgroups that is Councyl controls.

COUNCYL Decision Support Test Klant English (United Kingdom)

Choice advice | Model management | Notifications | User settings | Support

Add new choice
Enter the details on the choice task below and interpret the advice of Councyl based on these details afterwards.

Choice factors

- Age Unknown 30 70
- BMI Unknown <20
- Comorbidity Unknown Normal Very limited
- Capacity Unknown Limited
- Frailty Unknown 1 and 2

Choice characteristics

- Which choice did you make? NO YES
- How certain are you about your choice? Doubtful Super sure
- Did you make the choice based on a factor that is outside the above listed choice factors?
- Do you have additional comments?

Give me choice advice **79%**

Cancel Save choice

Figure 4.6: The interface for a clinical end user to enter choice specifications and request a choice recommendation in a dynamic BAIT-based CDSS.

Argument. Information cannot be retrievable to an individual clinical end user. With the implementation of Information specification Extensions, end users can request an update with a subgroup of choices (see subsection 4.3.1). For instance, the subgroup consisting of choices that senior physicians made. As a result, the updated model reveals the decision-making behaviour of that particular subgroup. Therefore, the architecture must ensure end users cannot link the subgroup to an individual end user. For instance, if there is only one senior physician in the decision-making context.

In addition, end users might want to reject the visualization of decision-making behaviour of subgroups that exist of slightly more individuals. For instance, an end user might argue that insight into the decision-making behaviour of two end users also violates privacy. Therefore, the architecture describes a data object representing the minimal number of clinical end users that a subgroup should represent. As a result, the CDSS counts the number of different clinical end users - different user IDs - in the choice set for each model update. Accordingly, the CDSS compares the counted number to the specified minimal number of clinical end users.

Consequently, a CDSS only presents the decision-making behaviour information when more clinical end users than specified by the minimal number of clinical end users data object made the choices. To beware of unaware clinical end users who want as much information as possible, the architecture describes Councyl as the entity managing the data object. By doing so, Councyl can always discuss the trade-off between the richness of the information that the CDSS can provide and the privacy consequences of setting the number with the main end user (for an explanation of end user types, see subsection 3.1.1). Because this data object is adjustable, Councyl can change the minimal number over time. For instance, when the pool of clinical end users changes.

4.4.5 Data accessibility for Council and clinical end users

Decision. The architecture describes a user authorization check as soon as it receives a request to present choice or decision-making behaviour information.

Argument. The outcomes resulting from a recommendation generator model update contains information on the model estimation and validation. However, the information does not have a unique receiver. Therefore, the architecture should describe the correct information flows to the intended receivers. Two reasons explain why the CDSS does not provide end users and Council with the same outcomes resulting from a model update. First, the performance metrics that the CDSS shares with both entities vary (see subsection 4.2.4). Second, end users may want the CDSS to hide the decision-making behaviour in their decision-making context for Council. With regard to the second reason, a CDSS developer needs to identify the preferences of clinical end users regarding the sharing of information. The information sharing options are as follows:

- Sharing both estimation and validation information with Council.
- Sharing only validation information that is per definition anonymous with Council.
- Sharing nothing with Council.

To declare the end users' preferences, the developer can use the Information Processing Agreement that Council already has in place. Next, the developer can implement the information flows to Council following the agreement. If the agreement states that Council cannot access any model update information, the developer will not implement the information flow with the outcomes resulting from a model update to the interface of Council.

The architecture describes an user authorization check process to check if the user ID requesting insight in the information is from Council or a clinical end user. Accordingly, the architecture ensures that a CDSS provides the correct set of update outcome information.

4.5 Summary chapter 4

The goal of chapter 4 is to present the structural specification of the dynamic BAIT-based CDSS architecture solution for Council. To this end, this chapter describes and arguments the nontrivial design decisions on the architecture solution. In total, the architecture stems from 17 nontrivial design decisions. The decisions concern either the architecture's structure, the processes for the estimation and validation of the choice recommendation generator model, the extensions of the architecture, or the management of information that a dynamic BAIT-based CDSS consumes and produces.

With regard to the structure, the architecture follows a layered and adaptable design so that the architecture is modifiable towards varying and changing preferences in different healthcare contexts. Moreover, the architecture guides the development of a dynamic BAIT-based CDSS without specifying any system software and hardware prerequisites.

An update of the recommendation generator model needs an activation. The architecture describes three extensions that allow customizing the level of updating automation. When activated, the update engine estimates a model with a "pooling" technique. By doing so, the update engine estimates the model parameters from experiment choices and real-life choices jointly. The estimation deals with missing attribute values by restricting the individual parameters to zero. The process to validate a new recommendation generator model follows a manipulated k-fold validation process. By doing so, the architecture ensures that a dynamic BAIT-based CDSS selects the choices used for the validation in a controlled way. As a result, the CDSS validates the model on the maximum amount of information that is available and only with unique, recent real-life choices.

To adhere to the varying preferences concerning the influence of particular choice types on an update of the recommendation generator model, the architecture defines three Information specification Extensions: end users can vary the weight of experiment and real-life choices (Extension 1), exclude real-life choices in terms of their features (Extension 2), and assign a different weight to a selection of real-life choices relative to the not-selected real-life choices (Extension 3). For the weight specification, end users can use an importance rating or importance balance. The first allows end users to specify a number that multiplies the influence of a particular set of choices

on an update. The latter allows end users to specify a percentage representing the share that a particular set of choices should have in the complete set information that an update incorporates.

Council implements an Information specification Extension as a user interface. Accordingly, end users can specify the preferred choice type selection and weighting settings via an interface. Because the extensions are implemented as interfaces, end users can always change the extension a CDSS operates without the need for any adjustments in the construction of the back-end design of the CDSS components. By letting clinical end users choose extensions, the CDSS is receptive to the input of clinical end users on the change of a dynamic BAIT-based CDSS.

At the start of an update of a recommendation generator model, the update engine checks the user settings regarding the choice information an update should process. These user settings inform the preparation of the temporary choice bases. These temporary choice bases contain the choice information that an end user wants an update to incorporate. If a CDSS is customized with one of the Information specification Extensions, this choice information forms a modified set of the original choices that a CDSS stores. By creating temporary choice bases at the start of each update, a CDSS will never lose the original content of the choice bases. By maintaining the access to the original choice information that is not modified according to any end user preferences, the inclusion of temporary choice bases enable end users to always specify a new set of choice information that should inform an update.

Finally, the architecture describes three components that protect sensitive choice information and decision-making behaviour information. The protection covers both the privacy of clinical end users within their decision-making context as well as against Council.

The decisions mentioned above form a basis for the architecture design for Council, which chapter 5 presents. As a result, this chapter represents a first step in answering the fourth sub-question:

4. What does a system architecture of a dynamic BAIT-based CDSS look like?

Chapter 5

Results: Architecture solution

The goal of this chapter is to present the architecture design that the architecture requirements outline (see subsection 3.2.3). The architecture forms the solution to the problem that Council encounters (see section 3.1). Section 5.1 presents a high-level overview of the architecture solution. The remainder of this chapter gives a detailed insight into the five components that the architecture describes: the adaptive choice base (section 5.2), the model quality monitor (section 5.3), the model update (section 5.4), the user settings component (section 5.5), and the model management module (section 5.6). The chapter concludes with a summary in section 5.7. The explication of all the architecture components answers the fifth sub-question:

5. What does a system architecture of a dynamic BAIT-based CDSS look like?

5.1 Architecture overview

A dynamic BAIT-based Clinical Decision Support System (CDSS) architecture specifies the components a BAIT-based CDSS needs to update according to contextual change. The core of the architecture includes five main components: an adaptive choice base, a model update engine, a model quality monitor, user settings component, and a model management component. The architecture connects all five parts to interfaces, which cover the Human-Computer Interaction (HCI). Business processes trigger these interfaces. The business processes are processes the CDSS stakeholders - Council and the clinical end users - perform to interact with a CDSS or complete tasks related to a CDSS. Therefore, the business processes represent (a sequence of) business practices that aim at achieving a specific result.

The yellow boxes in the architectural visualizations in this chapter represent business processes. If the architecture does not specify that Council triggers the process this means that the group clinical end users operating in a particular decision-making context are the stakeholders performing the business processes. As subsection 3.1.1 explains, a BAIT-based CDSS has two types of end users. The architecture does not prescribe which type of end user should perform which process. The group of end users should decide who is responsible for which business processes or select a main user who takes care of and controls all processes of the dynamic BAIT-based besides requesting choice advice. The latter will every end user facing a complex task aim to do. By doing so, the architecture avoids the situation in which one end user automatically possesses all power over the CDSS.

Most of the components consist of software processes and data objects. The blue boxes in the architectural visualizations in this chapter represent these software processes and data objects. The software processes describe the internal behaviour of an architecture component. A data object represents information the components processes consume or produce.

The architecture combines all business processes as a business layer and all software processes and data objects as an application layer. Because the elements at the business layer rely on the elements at the application layer, the layers follow a hierarchical organization. By distinguishing these different layers, the architecture follows a layered approach as section 4.1 explains. A layered approach commonly distinguishes three layers: the business, application, and technology layer ((Arsanjani, 2004; Lu, 2005)). However, the architecture is independent of hardware and software prerequisites. Therefore, the architecture does not describe elements at a technology layer.

5.1.1 The relation with the current static BAIT-based CDSS

A component that already exists in the static BAIT-based CDSS accompanies the five components in the architecture of the dynamic BAIT-based CDSS: the choice recommendation generator (Figure 5.2). The choice recommendation generator has two main functions. The first function is generating choice recommendations for end user's choice tasks using the choice recommendation generator model. This model represents the model a CDSS uses to generate recommendations. The second function is measuring and storing new real-life choices that clinical end users enter in the adaptive choice base. The architecture of the dynamic BAIT-based CDSS includes this component because the measurement of new real-life choices is key for a dynamic BAIT-based CDSS. Moreover, the recommendation generator in the architecture is slightly different from the generator the static BAIT-based CDSS uses (see subsection 4.4.1). As a result, the architecture solution describes six components. Figure F.1 presents a high-level overview of the architecture (Appendix F provides a larger version of this high-level overview). Appendix E presents the complete architecture. Appendix C contains the legend defining all components and shortcut words in the architecture.

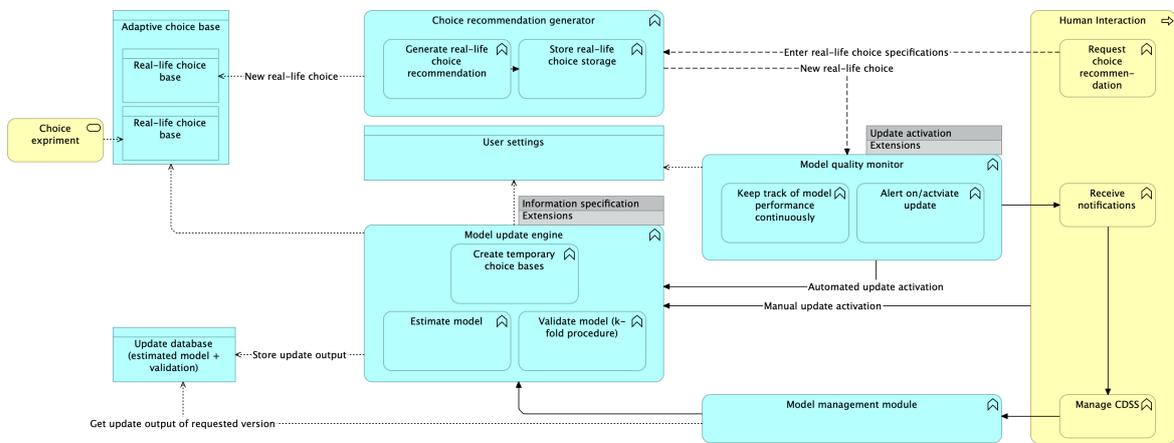


Figure 5.1: High overview of the dynamic BAIT-based CDSS architecture for Councilyl with six components.

consisted of

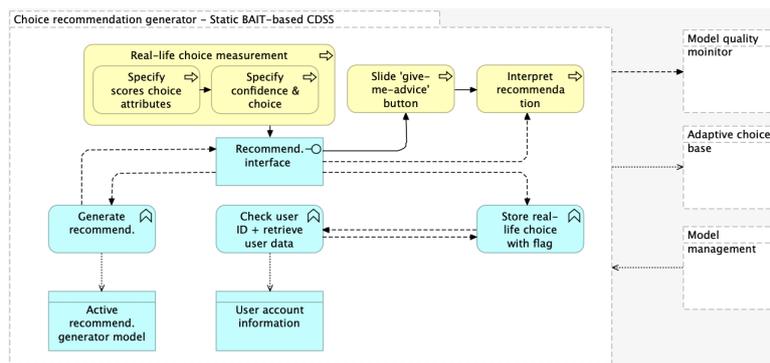


Figure 5.2: Architecture solution: Choice recommendation generator and relationships.

5.1.2 Relations between the components of a dynamic BAIT-based CDSS

The six key components of the architecture are interdependent. Every time a clinical end user enters a real-life choice, the adaptive real-life choice base stores the real-life choice. Moreover, the model quality monitor checks whether the recommendation of the CDSS aligns with the choice of the end user. Next, the monitor checks if the performance has declined below a specified level of acceptance. If so, this indicates that the recommendation generator model needs an update. The

model update engine estimates a new recommendation generator model and validates its performance. As a final step, the update engine replaces the recommendation generator model of the recommendation generator component with the new model. The user settings component captures all end user-specific preferences concerning the behaviour of all other components.

The six key components form the core of the architecture. The core covers the fundamental necessities that are time and context-independent (Fong, 2001; Madura, 2006; Moore & Chang, 1980; Yeung & Hall, 2007). Besides the core components, the architecture describes optional components that form extensions of the core. These extensions form “expanding subsets of system capabilities based on an initial nucleus of extensible features” (Moore & Chang, 1980, p. 12). By doing so, these extensions allow customizing a CDSS so that the CDSS matches the end user’s preferences (Moore & Chang, 1980). The main end user that subsection 3.1.1 introduced will have the responsibility for selecting extensions. The architecture describes two types of extensions. The grey frames in the architectural visualizations in this chapter mark the extensions. The architecture describes three extensions for the activation of the model update engine. Section 5.3 explains these extensions. In addition, the architecture describes three extensions for the incorporation of choices in a model update. Section 5.4 explains these extensions.

5.2 Adaptive choice base

The adaptive choice base consists of an experiment choice base and a real-life choice base. Figure 5.3 presents the subcomponents of the adaptive choice base. Because the choice base is adaptive, the choice base allows to add, delete or replace choices over time.

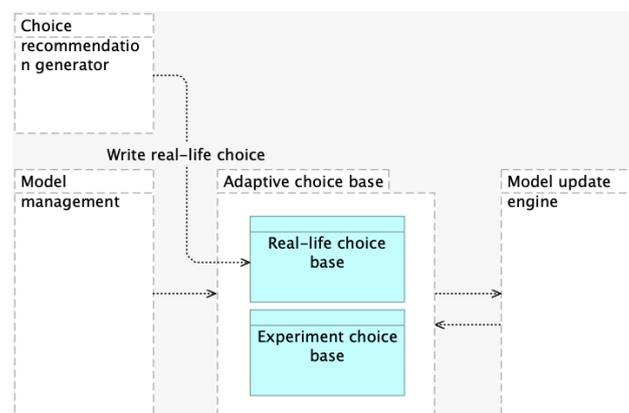


Figure 5.3: Architecture solution: Adaptive choice base module and relationships.

The addition of choices in each choice base is different. The adoption of new experiment choices requires manual action. Experiment choices stem from a choice experiment (see subsection 1.2.2). Therefore, new experiment choices require a new choice experiment. Council can replace the complete set of experiment choices in the adaptive experiment choice base with a new set of experiment choices. The features of an experiment choice are the choice maker, the scores on the choice attributes, the date on which an end user made the choice, and the time at which an end user made the choice.

End users add real-life choices continuously. An end user enters a new real-life choice via the interface that the end user also accesses for choice recommendation requests (see section 5.1). While entering a real-life choice, the end user specifies the following choice features: scores on choice attributes, choice recommendation of the CDSS, choice of the clinical end user (which is either in favour or against the treatment), the confidence of the clinical end user, and whether the clinical end user let an external factor influence the choice made. The choice base automatically stores information that is not choice specific, like e-mail address of the choice maker, the experience level (senior or junior), the discipline of the choice maker, the current date, and the current time. To this end, the choice recommendation generator checks the authorization of the clinical end user who entered the choice and retrieves the personal information stored for the specific end user. By doing so, the architecture ensures that the CDSS asks the end user only about choice-specific features and does not bother the end user with entering redundant information each time the user enters a

new real-life choice. The choice base stores the entered real-life choice with a flag. After a model update, the choice base removes all flags. As a result, this flag enables the CDSS to recognize which choices are new since the latest model update. Accordingly, the choice base informs the update engine on the choices that the engine can use for the model performance validation. The validation can only include the most recent real-life choices in the choice base (see section 3.2.3).

5.3 Model quality monitor module

The model quality monitor module has two main tasks. Figure 5.4 visualizes the subcomponents that realize these tasks. The grey boxes on the right represent the three extensions that allow customizing the activation of the update engine.

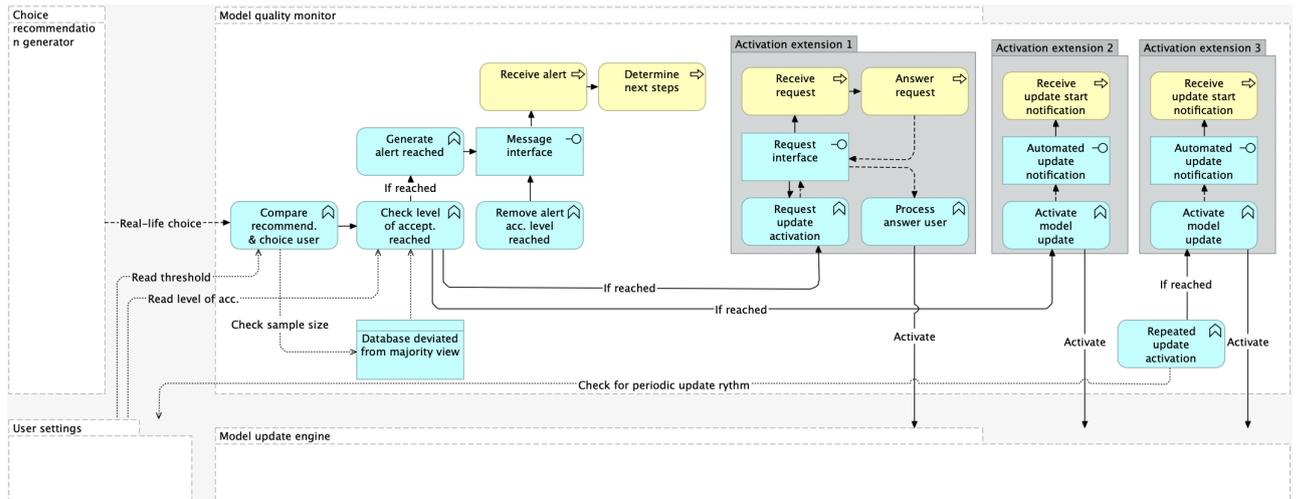


Figure 5.4: Architecture solution: Model quality monitor module and relationships.

The first main task is continuously assessing the performance of the choice recommendation model that the CDSS operates. Every time an end user adds a new real-life choice, the monitor checks if the recommendation model corresponds with the end user’s choice. To this end, the monitor first compares the recommendation of the CDSS with the majority threshold in the user settings (for an explanation, see section 5.5). The threshold is the minimum percentage that a recommendation should present for the end user to perceive the recommendation as a convincing vote in favour of the treatment. By doing so, the monitor categorizes the recommendation as a vote in favour or against the medical treatment. Next, the monitor checks whether the categorized recommendation corresponds with the end user’s choice (which is also either in favour or against the treatment). The monitor stores the real-life choices for which the recommendation deviates from the end user’s choice in a separate choice base. Clinical end users can always request the choices in this choice base. However, the CDSS will only present the choices that end users made themselves (see subsection 4.4.3).

The second main task is warning clinical end users on a decline in the performance of the recommendation generator model. The monitor warns end users as soon as the total number of deviating recommendations exceeds the level of acceptance that end users specified in the user settings. This warning remains visible until the update engine has completed the new update. In addition, the monitor resets the data object, keeping track of the number of deviating choices, to zero after the update engine completed the update.

If the CDSS contains an Activation Extension, the monitor has a third task. Depending on the implemented extension, this task includes either triggering an end user to activate the model update engine with an update request or directly activating the model update engine. The monitor directly activates the update engine as soon as the CDSS reaches the level of acceptance or if the periodic update rhythm in the user settings requires the activation of an update. Subsection 4.2.1 provides additional information on the Activation Extensions.

5.4 Model update engine module

The update engine consists of three main processes: the construction of the temporary choice bases (see subsection 5.4.1, a model estimation (see subsection 5.4.2), and a model validation (see subsection 5.4.3). The activation of an update triggers the construction of the temporary choice bases (see section 5.3 for the different activation variants). The fulfillment of the temporary choice bases triggers the latter two processes. The model estimation and model validation happen in parallel. Figure 5.5 visualizes these three processes.

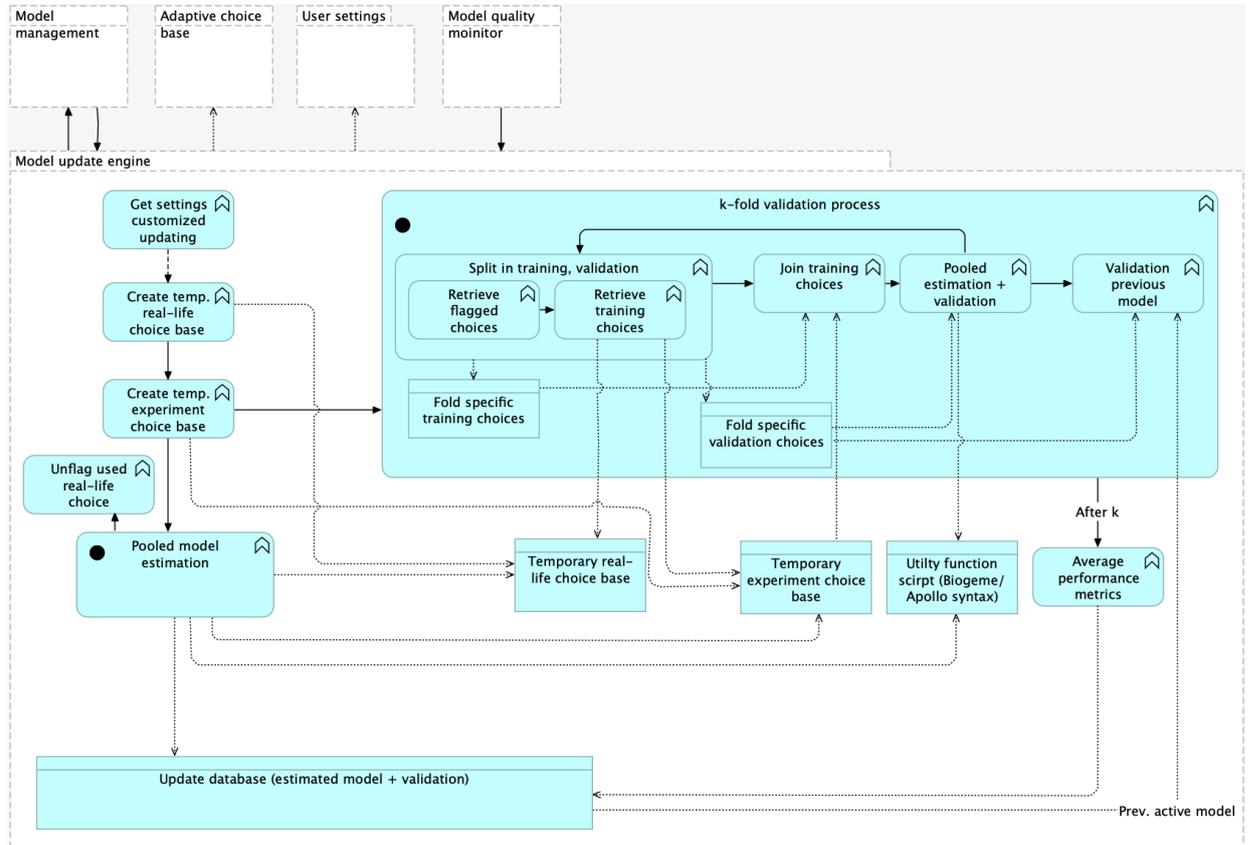


Figure 5.5: Architecture solution: Model update engine and relationships.

5.4.1 The construction of temporary choice bases

The temporary choice bases contain the choices that directly inform the estimation and validation of the model update. Without an Information specification Extension, the temporary choice base contains the same choices as stored in the adaptive choice base. However, if the CDSS includes an Information specification Extension, the temporary choice bases will include a specific set of choices.

The different extensions either allow clinical end users to vary the weight of experiment and real-life choices (Extension 1), exclude a set of real-life choices (Extension 2), or assign a different weight to a selection of real-life choices relative to the other real-life choices (Extension 3) (see subsection 4.3.1). At the start of an update, the engine checks the customized updating specifications in the user settings. Given the specifications, the engine picks and multiplies the correct experiment choices and real-life choices, and stores the resulting choice sets into the temporary choice bases (see Figure 5.5). As a result, the temporary choice bases may contain a sub-selection of particular choice types, a multiplied set of choices or both.

For Extension 1 and 3, the end user can choose between two weight specification variants: an importance rating (for instance, specifying that an update should include experiment choices as two times more important than real-life choices) or an importance balance (for instance, specifying that the update engine should inform the model with experiment choices for 70% and with real-life choices for 30%). A CDSS can contain all Extensions. However, the end user should choose between the weight specification options. Figure E.2 presents the Information specification Extensions as

the architecture describes them.

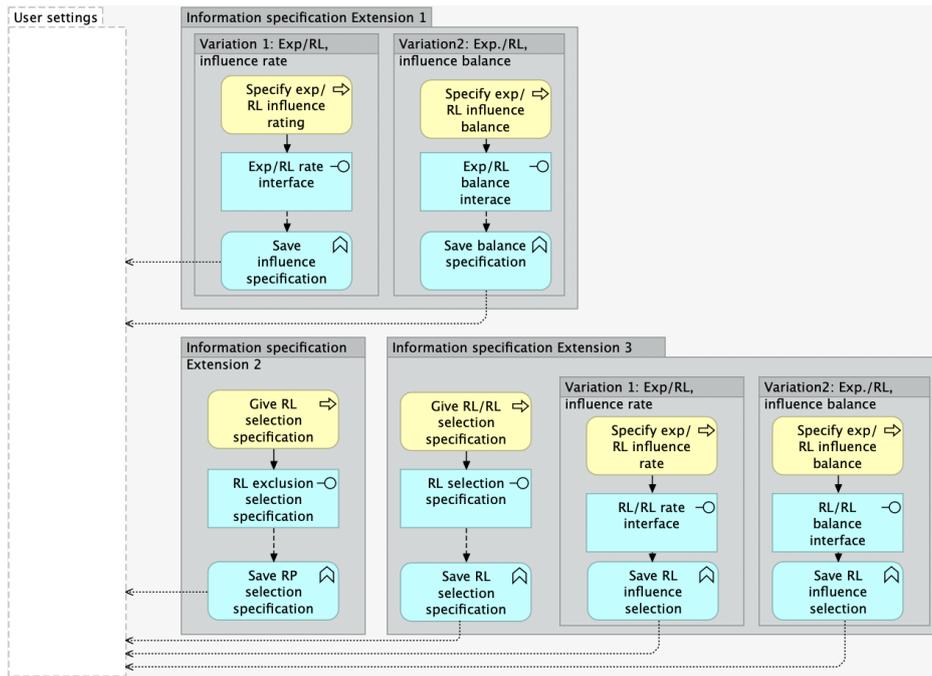


Figure 5.6: Architecture solution: Information specification Extensions 1, 2 and 3, and relationships (Exp. = experiment choices, RL=real-life choices).

5.4.2 The choice recommendation generator model estimation

The model estimation aims to estimate a new choice recommendation generator model with new real-life choices end users added. The model estimation process takes the choices in the temporary experiment choice base and real-life choice base. By combining these experiment choices and real-life choices, the estimation process estimates the parameters for all choice attributes. The estimation process includes choice attributes for which information is missing as if they did not influence the clinical end user's choice (see subsection 4.2.3). After the completion of an update, the update engine empties the temporary choice bases, replaces the previous active recommendation model with the newly estimated model, and adds the estimation outcomes to the database with model updates.

5.4.3 The choice recommendation generator model validation

The goal of the model validation is to assess the performance of the newly estimated recommendation generator model. The model validation uses a manipulated k-fold validation process (for an explanation, see subsection 4.2.5). The update engine estimates k models during this process, each based on a slightly different choice set. End users can change the value for k in the user settings (see section 5.5). To avoid confusion with the model estimation that provides the new recommendation generator model in subsection 5.4.2, the choice set used for the k model estimations as part of the validations refers to the training set. For each of the k estimated models, the update engine determines the performance.

To estimate and validate k models, the update engine constructs k training sets and validation sets. This construction does not happen entirely randomly. The validation set only contains real-life choices from the original real-life choice base (instead of the temporary choice base). By doing so, the validation set contains no duplicates (the temporary choice bases may contain multiplied and thus duplicated choices). The training set used for the estimation only contains values that are not in the validation set.

To determine the performance of each model, the update engine generates a choice recommendation for each choice in the validation set. The update engine determines how well the recommendations correspond with the real-life choices of end users in terms of the performance

metrics (see section 2.2.2). After the final iteration, the update engine averages the metrics resulting from all k iterations. The result is together with the estimated model stored in the database with model updates. The update engine informs clinical end users with the Correspondence rate and the Agreement table (see subsection 4.2.4). Because information on model performance is anonymous, the update engine informs Council with all performance metrics listed in section 2.1.1 if the Information Processing Agreement does not obstructs this.

To finish, the architecture describes that the update engine also assesses the performance of the previous recommendation generator model on the new real-life choices. By doing so, clinical end user can check if the update forms an improvement compared to the previous update given the developed decision-making context.

5.5 User settings component

The user settings component captures the specific preferences of end users in a healthcare decision-making context regarding specific values and mechanisms with which the CDSS operates. Figure 5.7 presents the subcomponents of the user settings component.

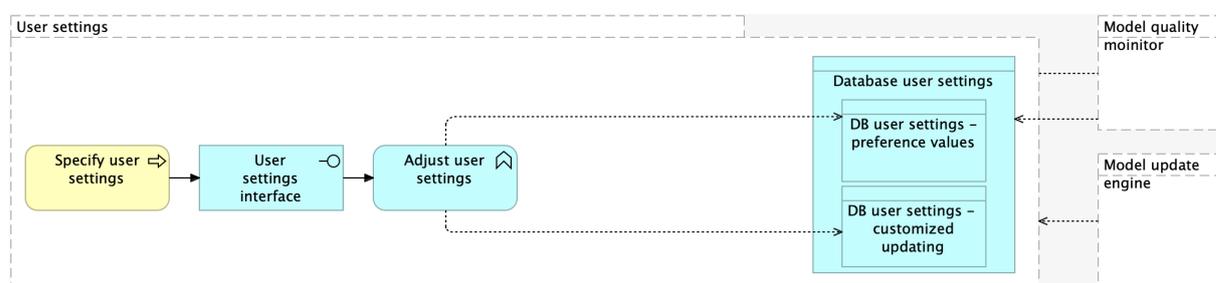


Figure 5.7: Architecture solution: User settings component and relationships.

The user settings component consists of two parts. One part of the user settings includes the preferences values. End users have to specify the following values:

- Value for k : the number of repetitions in the k -fold validation process. The higher the value for k , the more model estimations the update engine validates before it calculates the performance metrics.
- Value for majority threshold: the percentage above which the end user considers the recommendation as a convincing yes in favour of the treatment.
- Value for the level of acceptance: the number of real-life choices for which the dynamic BAIT-based CDSS is allowed to give a recommendation that deviates from the clinical end user's choice.
- The additional action associated with the level of acceptance: the action that the model quality monitor must undertake as soon as the CDSS reaches the specified level of acceptance. Two of the Activation Extensions (see subsection 4.2.1) represent the alternative actions: automatically activate a model update or send a model update request to the end user.
- Periodic update: the options to install a repetitive update activation. The end user has two options. The end user can specify a number of new choices entered. For instance, by letting the model quality monitor activate the update engine every ten new real-life choices. End user can also specify a period. For instance, by letting the model quality monitor activate the update engine every 30 days. This user setting represents the third Activation extension (see subsection 4.2.1).
- Minimal subgroup value: the minimal number of clinical end users a subgroup must represent before the CDSS can give insight into the decision-making behaviour of the subgroup (see subsection 4.4.4).

The second part of the user settings component is only relevant when a CDSS includes an Information specification Extension. This part stores the specifications of clinical end users concerning

the influence of particular choice types on updates. With the Information specification Extension 1, clinical end users can specify a weight for experiment choices relative to real-life choices via either the importance rating or balance. With the Information specification Extension 2, clinical end users can specify a selection of real-life choices. With Information specification Extension 3, clinical end users can specify a selection of real-life choice and a weight for the selected choices via either the importance rating or balance (see subsection 4.3.1).

Council will need to guide the first specification of the user settings. By doing so, Council can make end users aware of the effects of the various settings. Therefore, the architecture provides both clinical end users and Council with access to the user settings of a CDSS. The user settings do not contain privacy-sensitive information. If clinical end users consider changing in the settings later in time, end users can consult via the support component (see section 5.6). Moreover, the user settings interface provides buttons that explain the meaning of all user settings.

5.6 Model management module

The model management component includes all subcomponents that allow keeping track of and controlling a CDSS. The model management component supports tasks of both clinical end users and Council. Figure 5.8 presents all model management subcomponents.

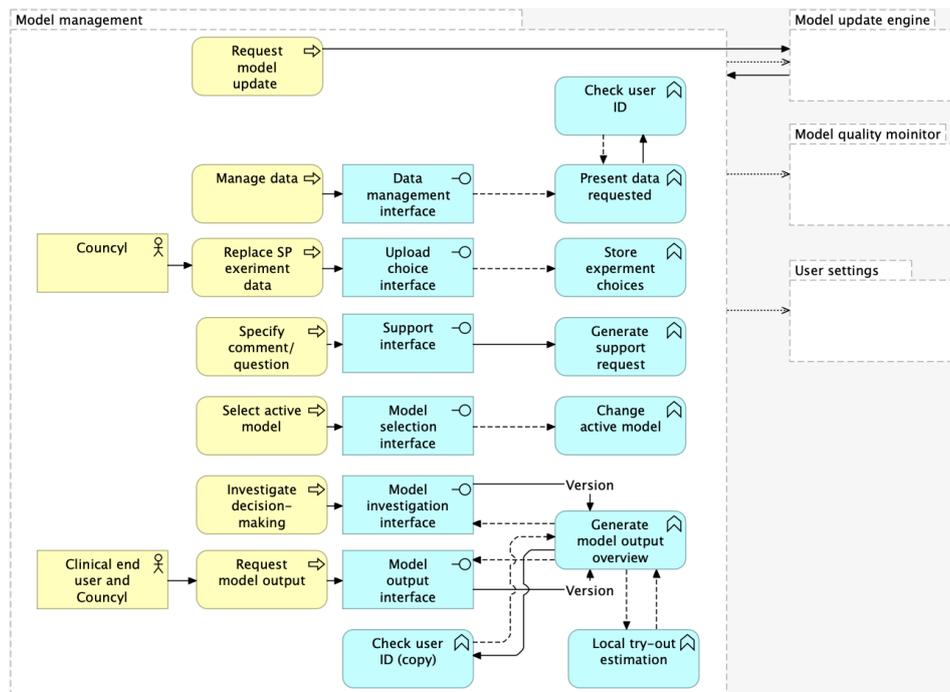


Figure 5.8: Architecture solution: Model management component and relationships.

The component supports five main tasks of end users. First, end users can manually activate a model update. As soon as the update engine completed the model update, the model management component provides feedback in the form of a brief update confirmation. This confirmation also requests the end user to view the update outcomes. Second, the model management component enables clinical end users to access choices stored in the adaptive choice bases or the database with deviating choices (see section 5.2). The model management component only presents choices the end user requesting the choices made. To this end, the data presentation process first checks the user ID of the requesting end user. Next, clinical end users can request support from Council via the management module. For instance, if the outcomes resulting from a model update are unclear or an end user experiences problems with a CDSS's functionalities. Fourth, clinical end users can always reset the active recommendation generator model to a previous model update if they perceive the new model update as undesired. This functionality also allows end users to try out a new update with new real-life choice information the CDSS collected so far. For instance, to check if the performance would already improve. Fifth, clinical end users can always request the outcomes resulting from a model update - including the estimated model parameters and the

model validation performance in terms of the Correspondence rate and the Agreement table (see subsection 4.2.4) - of the current and all previous model update versions.

Finally, clinical end users can investigate the decision-making behaviour within the pool of clinical end users over time. In addition, end user can request insight into the decision-making behaviour of particular subgroups of physicians. For instance, end users can investigate if senior clinical end users make choices differently than juniors or differently than they did six months ago. Similar to Information specification Extension 2 and 3, end users can specify subgroups in terms of the choice features (see subsection 4.3.1). The subcomponent realizing this decision-making investigating is the local estimation component. This component can estimate a choice model that makes the decision-making behaviour of clinical end users visible. However, the local estimation component does not activate the process that stores estimation outcomes or the process that resets the active recommendation generator model. Accordingly, the estimation in this environment does not influence the operation of the CDSS. To avoid privacy issues, the local try-out estimation process always checks if a requested subgroup does not harm the privacy of clinical end users before it estimates a choice model (subsection 4.4.4). To this end, it compares the number of user ID's in the requested subgroup with the minimal subgroup substance in the user settings (see section 5.5).

The model management module supports Council with two processes. Similar to clinical end users, Council can also request the outcomes resulting from model updates of a CDSS in a particular healthcare context. However, the model management component provides Council with a selected set of information (see subsection 4.4.5). To provide Council with the correct information, the model management module always checks the authorization of the entity requesting the information. Secondly, Council can replace the experiment choices in the experiment choice base. Because experiment choices originate from a controlled experiment (see subsection 1.2.2), Council can only replace all experiment choices in a CDSS's experiment choice base with a complete set of new experiment choices (for a detailed explanation, see subsection 2.2.1).

5.7 Summary chapter 5

The goal of chapter 5 is to present the dynamic BAIT-based CDSS architecture designed for Council. The design follows the architecture requirements that subsection 3.2.3 identified and includes all architecture specifications that chapter 4 presents.

The architecture defines six main dynamic CDSS components. All components represent business (organizational processes) or application (software processes and data objects) components. The components do not restrict a CDSS developer with any software and hardware prerequisites. The first main component is the choice recommendation generator. This generator is part of the static BAIT-based CDSS. The dynamic BAIT-based CDSS architecture includes a choice recommendation generator component for the measurement of new real-life choices. When an end user enters a real-life choice to get a recommendation from the CDSS, the end user can save this choice. Subsequently, the recommendation generator component stores the real-life choice in the adaptive choice base. The adaptive choice base forms the second component and consists of an experiment choice base and a real-life choice base. The third component is the model quality monitor that continuously assesses the performance of the choice recommendation generator model that a CDSS operates. Moreover, this monitor warns clinical end users as soon as the performance has declined below the level of acceptance that end users specified in the user settings.

The fourth component is the model update engine that realizes the updates of the choice recommendation generator model. To this end, the engine runs three processes: constructing temporary choice bases, estimating a new recommendation generator model, and validating this recommendation generator model in terms of performance metrics. The fifth component that the architecture defines is the user settings component that captures the specific preferences of clinical end users regarding a dynamic BAIT-based CDSS. Finally, the model management component allows end users to keeping track of a CDSS and to control a CDSS. The architecture describes interfaces for all dynamic CDSS components. These interfaces facilitate the business processes that the architecture includes. All interfaces together cover the Human-Computer Interaction between a CDSS and clinical end users or Council.

Chapter 6 presents the evaluation of the architecture solution for Council. The description of all the architecture components answers the fourth sub-question:

5. What does a system architecture of a dynamic BAIT-based CDSS look like?

Chapter 6

Evaluation of the architecture solution

This chapter presents the evaluation of the architecture solution. Architecture evaluation and refinement activities took place throughout the research. This ongoing process of evaluation and refinement resulted in the architecture presented in chapter 5. This chapter aims to evaluate this final architecture to assess whether the formulated architecture requirements together guide towards the design of a valuable architecture for Councilyl. Section 6.1 explains the evaluation approach. Section 6.2 and section 6.3 together explicate the evaluation of the architecture. As a result, this chapter answers the fifth sub-question:

5. To what extent does the designed architecture form an effective tool for guiding the development of dynamic BAIT-based CDSS?

6.1 Evaluation Approach

This section presents the evaluation approach. The evaluation in the context of Action Design Research (ADR) consists of two processes. The first process runs throughout the design process to identify weaknesses and areas of improvement and refinement for an artefact under development (Venable, Pries-Heje, & Baskerville, 2012). By doing so, this research evaluates the early designs in a formative manner. The evaluation in the final cycle in ADR is rather summative and involves an evaluation of the final design (Venable et al., 2012; Sein et al., 2011). This type of evaluation aims to determine the utility and efficacy (or lack thereof) of an artefact for achieving its purpose (Venable et al., 2012). As a result, the summative evaluation is the process of establishing how well the architecture performs given its purpose (March & Smith, 1995). A disadvantage of the summative approach is that the requirements cannot form the basis of adequate architecture design guidelines when the evaluation indicates that the architecture is low in utility. However, this risk is negligible because the final architecture is the result of an ongoing building and evaluation process that closely involved Councilyl (Sein et al., 2011). Accordingly, it is unlikely that the architecture is not in line with its intended purpose.

The purpose of the architecture is to guide dynamic BAIT-based Clinical Decision Support Systems (CDSS) for different healthcare decision-making contexts. The assessment of how well the architecture performs in doing this requires insight into the architecture's usability and usefulness for a CDSS developer. CDSS architecture design literature proposes several methods to evaluate architectures. The two most common methods mentioned are the scenario-based architecture evaluation and prototyping- or simulation-based architecture evaluation (Mårtensson, 2006). The first uses a scenario profile which forces a concrete description of the non-functional quality requirement (Lassing, Bengtsson, Van Vliet, & Bosch, 2002). Next, designers use the scenarios to go over all components of the architecture and assess the likely effects on the system instance (Kazman et al., 1998; Kazman, Bass, Abowd, & Webb, 1994). The designers document all imaginable consequences. The prototyping-based method focuses on evaluating technological components that the architecture describes in the intended run time environment (Martensson, Grahm, & Mattsson, 2004). By doing so, designers can make accurate measurements of the intended system before they

completely build the system (Martensson et al., 2004).

Unfortunately, well-nigh all the architecture evaluation methods that literature proposes concern the system rather than the architecture that describes the system (Byrnes & Kyrtzoglou, 2006; Folmer, Van Gorp, & Bosch, 2004; Shanmugapriya & Suresh, 2012; Patidar & Suman, 2015; AlSharif, Bond, & Al-Otaiby, 2004; Stevanetic & Zdun, 2015; Babar, Winkler, & Biffi, 2007). These methods help system designers with finding lacks in the system design early on in the software development life cycle. As a result, the methods focus on assessing the extent to which the system as the architecture describes it will be useful and usable. Although the architecture must describe the components that generate the desired CDSS functionalities, the evaluation part of this research does not aim to solely investigate whether the CDSS as an artefact is useful and usable for Councilyl. Instead, the evaluation aims to investigate whether the architecture as an artefact is useful and usable for Councilyl.

To cover both aspects - whether the architecture describes the components that generate the desired functionalities and whether the architecture itself is useful and usable - the evaluation is twofold. The first part of the evaluation focuses on the architecture as a static artefact. Static refers to the aspects of the architecture that one can observe without the implementation of the CDSS that the architecture describes (Knodel, Muthig, & Naab, 2006). In essence, this is the architecture as a blueprint for CDSS developers. This static-oriented evaluation includes a review session with Councilyl to assess whether Councilyl perceives the architecture as useful and usable (see section section 6.2). Because the usefulness of the architecture also depends on whether the components that it describes constitute the desired CDSS functionalities, the evaluation is expanded with a dynamic assessment. Dynamic refers to the architecture aspects that one can observe during a CDSS's run time. The aim of the dynamic-oriented evaluation is to show the architecture components describe the desired functionalities in a feasible way and to highlight any pitfalls (Rozanski & Woods, 2012). The dynamic-oriented evaluation includes the evaluation of a test execution of the main components that the architecture describes and of a series of mock-ups (see section 6.3).

6.2 Static-oriented evaluation

This section describes the method and results of the first part of the evaluation: the static-oriented evaluation. The static-oriented evaluation consists of a session with Councilyl to determine the usefulness and usability of the architecture as a tool to guide the development of dynamic BAIT-based CDSSs. Usefulness indicates the value of the architecture Councilyl ascribes to it. Usability measures the ease with which a user completes a particular task while using the artefact.

In the context of an architecture, usability essentially comes down to the understandability of the architecture. In the end, the main purpose of the architecture is to abstract away small details and give a clear overview of the main components of the dynamic BAIT-based CDSS (Oreizy et al., 1999; Stevanetic & Zdun, 2015). Understandability refers to the extent to which the architecture is understandable to Councilyl's CDSS developers (Alenezi, 2016). Understandability of the architecture is important because an architecture that is not understandable makes it is simply impossible to develop the CDSS as intended (Akour, Aldiabat, Alsghaier, Alkhateeb, & Alenezi, 2016; Alshammary & Alenezi, 2017; Alenezi, 2016). Moreover, the understandability determines how reusable and maintainable the architecture is for Councilyl (Akour et al., 2016; Stevanetic & Zdun, 2015).

To guide the static-oriented evaluation session with Councilyl, the two concepts usefulness and understandability, were broken down into measurable concepts. The evaluation questions to assess the usefulness included the following concepts: the guidance capability for CDSS development, completeness, expected development time, stability, and competitive ability of the architecture. Table 6.1 presents the usefulness concepts, the questions asked to Councilyl and a summary of the answers Councilyl provided. In addition, the sessions assessed what Councilyl finds the most valuable aspect of the architecture. The evaluation questions to assess the understandability included the following concepts: complexity, clarity of components defined, and consistency of the architecture description. Table 6.2 presents the understandability concepts, the questions asked to Councilyl and a summary of the answers Councilyl provided.

The findings summarized in Table 6.1 and Table 6.2 indicate that Councilyl finds the architecture useful and understandable. Additional insights are as follows. Councilyl declared that developers might deem the adaptable architecture incomplete if new user preferences occur over time. How-

ever, the developer of Council is confident that the architecture can be easily adjusted to cover these preferences. Moreover, a developer might require additional supporting tools. These tools store the knowledge about the architecture in the long term and inform new developers, who will join Council in the future and were not part of the architecture design process. Council declares to perceive the following guiding tools as valuable: an overview of all data objects related to the architecture components as Appendix J presents, a high-level overview of the architecture as Appendix F presents, a series of guidelines for the implementation of the architecture as Appendix I presents, and a legend representing all elements in the architecture as Appendix C presents. Finally, Council preferred to have an overview of all design decisions, and argumentation of these decisions as chapter 4 presents.

Table 6.1: *Evaluation of usefulness*

Focus area	Questions asked	Findings
Guidance capability	To what extent can the architecture guide the development of a dynamic CDSS for different types of end users with different preferences	The architecture provides a valuable and effective starting point for the development.
Completeness	Are aspects that have to be dealt with missing?	No important aspects are missing. Beyond this research, however, it might be valuable to investigate what other software packages than currently used could be used for the model estimation. Additionally, over time extreme scenarios may occur for which adjustments are needed. However, it can be stated with confidence that the architecture allows for these adjustments.
Acceptable development time	Are you confident that you can develop a system for a client in a proper amount of time?	Yes. For the first implementation, however, a bit of extra time might be needed. This is not perceived problematic, since it is not expected that the time needed will exceed what is considered reasonable.
Acceptable development time	Are you confident that you can develop a system for a client in a proper amount of time?	Yes. For the first implementation, however, a bit of extra time might be needed. This is not perceived problematic, since it is not expected that the time needed will exceed what is considered reasonable.
Stability	Given the adaptable approach, do you feel confident that the architecture forms a stable starting point that can be adjusted for use over time?	Yes.
Competitive ability	What are alternatives for the architecture you would rather use?	None.
Most valuable specification	What are the most valuable aspects of the architecture?	The explication and tested codification of all potential extensions that may be preferred by different end users in different healthcare decision-making contexts.

6.3 Dynamic-oriented evaluation

This section describes the method and results of the second part of the evaluation: the dynamic-oriented evaluation. In contrast to the static-oriented evaluation, the dynamic-oriented evaluation focuses on the run time behaviour of the system components defined by the architecture. The dynamic-oriented evaluation aims at assessing whether the architecture describes the components that generate desired CDSS functionalities, ascertaining whether the risky components of the architecture are feasible, and investigating potential problem patterns or risks (Mårtensson, 2006; Rozanski & Woods, 2012). The dynamic-oriented evaluation consists of two parts: the evaluation

Table 6.2: *Evaluation of understandability*

Focus area	Questions asked	Findings
Perceived understandability	Do you feel you understand the architecture?	Definitely. However, this is also the result of the repeated evaluation sessions. For a person with another system development role, explaining guidelines would increase the reasoning behind and, therefore, the understandability of the architecture. Similarly, when it would be out of sight of time for a while, additional effort would be needed to understand everything at the same level. For a new person, the high overview is a nice first introduction.
Complexity	Do you feel the architecture is too simple or too complex for doing the job of developing a system?	It is complex, because it contains many components and relations. However, it is complex at the right level because if it would be less complex it lacks the definition of relevant aspects. This causes a decrease in the usefulness of the architecture.
Clarity of components	What do you think about the clarity of the components (like services, interfaces, process) and how they differ?	At first sight, the different layers provide clarity. For the specific types of components, a legend would be needed. With a legend, all types of components would be clear.
Clarity of components	Do you expect that additional research effort is needed for development because aspects are still unclear due to a too high level of abstractness?	No significant research efforts are expected, because the level of detail is just right. Some coding details will need to be specified by system engineers. This is, however, beyond the goal of the architecture.
Clarity of components	Does the architecture make a clear distinction between the core and the extensions that can be chosen given preferences of the end user?	Yes. Mainly because of the marked boxes and associated references (1A,1B, and 2A,2B,2C). It is clear what components are essential core elements and which function as optional extensions of the core. In addition, it makes clear which extensions can be implemented jointly, and which extensions are each others' substitutes. Moreover, the number of extensions is satisfying as well as manageable and comprehensible.
Consistency	Is the description architecture description perceived	Yes, because the architecture is defined using unique signs for all types of components. Because of this systematic translation of design choices, it is ever clear what an element represents.
Additional guidance	What additional tools would make the architecture better understandable?	A legend of all component representations, an overview of all data objects that are represented by the overarching databases, a brief reasoning of the extensions and in which contexts they are of value, and a brief reasoning of the design decisions so that the design is tractable. The textual explanation and visual architecture description then enhance each other.

of a proof-of-technology (see section 6.3.1) and the evaluation of a series of mock-ups (section 6.3.2).

6.3.1 Proof-of-technology

This section presents the evaluation of the proof-of-technology. A proof-of-technology is a temporary code or another form of implementation developed to prove a risky technological component in the architecture is feasible and effectively generates the desired functionalities (Rozanski & Woods, 2012). A proof-of-technology allows evaluating the technical decisions concerning the vital components in the architecture and understand the implementation technology of these components in a safe environment (Rozanski & Woods, 2012). The most vital and risky architecture components are four components that update a CDSS with new choices. These components are the core updating engine architecture and three Information specification Extensions that modify the information the update engine consumes (for an overview of these Extensions, see subsection 4.3.1). As a result, evaluating the proof-of-technology shows if the components as defined indeed create the temporary choice bases and if estimation process and validation process can use these temporary choice bases properly (see for an explanation of these processes section 4.3). Table G.1 gives an overview of the implemented components. The goal of the proof-of-technology is to assess whether Council believes that these four updating components in the architecture generate the intended updating functionalities and does not give rise to any risks.

Method

The proof-of-technology of this research consists of implementing and executing all the four updating components the previous section mentions. The architecture guided the implementation of the four components. Therefore, the proof-of-technology forms a reflection of the four architecture

components. Without the architecture as a guiding tool, the implementation would have been different. The list below briefly describes the five most prevalent differences:

1. The choices the implementation would use for the validation process would be incorrect. Instead of only incorporating recent real-life choices, the implementation would select a random set of choices from the set of available choices.
2. The implementation would not include a separate estimation process that parallels the k-fold validation process. Instead, the implementation would solely contain a k-fold validation process from which the update engine takes the average estimations to provide a new recommendation generator model.
3. The implementation of Tested component 1 (see Table G.1) would potentially include an estimation process that follows sequential technique (for an explanation, see subsection 4.2.2).
4. The implementation would not include a process that "unflags" the real-life choices used in an update. Consequently, the update engine would not distinguish the new real-life choices that the update engine needs to incorporate for validation from the choices that are not recent.
5. The implementations of Tested Component 2,3 and 4 (see Table G.1) would not contain temporary choice bases. Instead, the implementation would modify the original choice bases, which causes the loss of the original choice set (for an explanation, see subsection 5.4.1).

The implementation used a python package called Biogeme. This package relies on the Python data analysis library Pandas. It is an open-source package designed for the maximum likelihood estimation of parametric models in general with an emphasis on discrete choice models as needed for the implementation (Bierlaire & Fetiariison, 2009). It contains all the required methods to estimate choice models. In Appendix G the implementation scripts representing the proof-of-technology are listed. The implementation for a proof-of-technology is temporary and discarded after the evaluation (Rozanski & Woods, 2012).

The execution of the implemented component relied on test data. The test data set contains experiment choices and real-life choices collected in a healthcare context. The represented choice task is the choice in favour or against ICU uptake of COVID patients. The answer type was twofold: no ICU uptake and ICU uptake. The data set represents 502 choices: 425 experiment choices and 77 real-life choices. Physicians made the real-life choices in the period from March 2020 to October 2020. Dummy data enriched this set of choices. This dummy data contains the additional features of choices that the current static BAIT-based CDSS does not collect (for an overview of the choice features, see Table 4.1). Because the test data includes dummy data, it is meaningless to make and present claims about the values resulting from the execution. Instead, the goal of the proof-of-technology is to allow Councilyl to judge whether the components generate the intended functionalities and are feasible. Therefore, the evaluation of the proof-of-technology consisted of a software walk-through during which Councilyl assessed the functionalities and outcomes generated by the scripts. Appendix G presents the outcomes of all scripts central in the walk-through with Councilyl.

Results

Councilyl confirmed that the architecture describes the components that generate the intended updating functionalities feasibly. However, the walk-through pointed at a single required adjustment. The scripts presented in section G.1 and the architecture that in chapter 5 already capture this adjustment. The adjustment results from the manipulated data split in the k-fold validation that was incorrect for the type of choice data k-fold validation consumes.

In medical practice, multiple physicians decide on the treatment of a single patient. Consequently, multiple clinical end users enter their view on the patient as a new real-life choice into a BAIT-based CDSS. However, these real-life choices concern the same patient and are therefore relatively similar. When the k-fold validation ignores the dependence between these real-life choices, the k-fold validation will create a training and validation set without controlling for recurring data entries. As a result, the training set may contain multiple choices representing similar patients. At the same time, the validation data includes choices representing other patients. The trained model will be overfitted in favour of the patients over-represented in the training set, and the model will not represent the population of patients well (Huang, Hung, & Jiau, 2006). As a consequence, all

Table 6.3: *Overview table of the implemented and tested architecture components*

Reference	Component	in architecture	Explanation of component functionality
Tested 1	component	The model updating engine	The estimation and validation of a choice model with experiment and new real-life choices. The process part of this component are needed for every clinical end user. The estimation should result in a set of parameters for all choice attributes and an error term. The validation should indicate the performance on new real-life choices in terms of the Correspondence rate and the recommendation-choice Agreement table (also referred to as the Confusion Matrix).
Tested 2	component	Information specification Extension of updating engine 1	The estimation and validation of a DCM in which the importance of experiment and real-life choices are varied with both a importance rate and balance (see subsection 4.3.1). For instance, when a clinical end user values experiment choices two times more important than real-life choices.
Tested 3	component	Information specification Extension of updating engine 2	The estimation and validation of a DCM in which a particular set of real-life choices are excluded. For instance, when a clinical end user wants the CDSS to only incorporate choices made by senior physicians. The choice base that is used for the estimation should only include the selected choices.
Tested 4	component	Information specification Extension of updating engine 3	The estimation and validation of a DCM in which the importance of a particular set of real-life choices is differently weighted compared to the remaining real-life choices by using both an importance rate and balance specification. For instance, when a clinical end user considers choices made by senior physicians as more important than choices that were made by junior physicians.

scripts generated divergent performance metrics for the k estimated model. An additional manipulation of the k-fold validation equally divided the real-life choices that belong to the same patient over both the training set and the validation set. Because physicians have slightly different views on a patient, the training set and validation set will not include the exact same cases. After the modification, the scripts generated a stable model performance over the k estimations.

After the adjustment, Council declared to be confident that the architecture components represent a dynamic BAIT-based CDSS that end users can update with new choices. More specifically, the execution of component 1 shows that the architecture correctly describes a pooled model estimation with experiment and real-life choices. Moreover, the validation processes in the architecture generate the intended performance assessment insights. Finally, the validation only used the most recent real-life choices stored in the choice base. The outcomes generated by executing components 2, 3, and 4 showed that the temporary choice bases copy correct choices from the choice base. Moreover, the estimation and validation processes successfully used the choices from the temporary choice bases.

6.3.2 Mock-ups

This section presents the evaluation of the mock-ups. Mock-ups are full-scale models of a design that are useful for demonstration and evaluation of a design (Bayramzadeh et al., 2018). These mock-up evaluations involve testing various aspects of a proposed design. Evaluations of physical mock-ups form an effective tool to communicate the design of healthcare systems since they improve understanding and communication between healthcare providers and the designer (Keys, Silverman, & Evans, 2017). For that reason, designers of healthcare systems increasingly use mock-ups to support and validate design decisions (Bayramzadeh et al., 2018)

Method

The designed mock-ups demonstrate all functionalities that components in the architecture generate. Appendix H presents the complete set of the final mock-ups, including the improvements resulting from the findings that section 6.3.2 presents. The main goal of the mock-up evaluation is to assess whether the architecture covers the correct set of functionalities and whether essential features are missing or are superfluous. To this end, the evaluation of the mock-ups consisted of two steps. The first step is checking whether all aspects of the architecture are complete while designing the mock-ups. The second step is evaluating the mock-ups with all representatives of Council and a clinical end user who also participated in the interviews for the elicitation of requirements. Although mock-ups represent interfaces, the goal of the evaluation is not to assess the layout design of the mock-ups. However, Council and the end users should perceive the content as clear and effective enough to understand and use the functionalities.

Results

The mock-up evaluation confirms that the architecture covers the essential functionalities of a dynamic BAIT-based CDSS. Besides this confirmations, the evaluation with both Council and the end user pointed at a number of required adjustments to the architecture design. The list below summarizes the four most prominent adjustments.

Findings session with Council. Council made the two following suggestions on the mock-up designs during the mock-up evaluation session.

1. The architecture should protect the privacy of end users within the healthcare organization. At first, the architecture dealt with privacy as an issue involving two stakeholder parties: Council as an external CDSS provider and the healthcare context as end user. The evaluation with Council showed that privacy is also an issue among clinical end users within the healthcare context. Consequently, the architecture needs two adjustments to ensure that the architecture does not guide towards CDSS's that visualize choices and decision-making behaviour information in a way that harms the privacy of clinical end users within the organization. The first adjustment is that the architecture only allows end users to access choices they made themselves. The second adjustment is that the architecture ensures that insight in decision-making behaviour is never retrievable to an individual clinical end user.
2. The architecture should allow end users to exclude choices from updates with recommendations for which the CDSS was confident but deviated from the end user's choice. The architecture already allowed end users to request the selection of choices for which the recommendations deviated from the end user. During the mock-up demonstration, Council pointed out that the architecture should also allow end users to tune updates of the CDSS by excluding the choices with recommendations about which the CDSS was confident but deviated from the end user. These choices refer to the situation in which an end user made a choice significantly different than the majority of his or her colleagues. By doing so, end users can, for instance, first investigate why the end user made a significantly different choice than the CDSS recommended before they let an update incorporate this type of choice. Ensuring the architecture describes a CDSS that distinguishes these significantly deviating choices required making two adjustments. The first adjustment involved changing the architecture data objects in the data overview by adding confidence as a feature of real-life choices (see Appendix J). The second adjustment involved adding a selection choice feature with which end users can specify the choices they want to include in a model update. By doing so, end users can make more sophisticated subgroups of choices for a model update or for investigating the decision-making behaviour of end users in the healthcare decision-making context (see 5.6).

Findings session with clinical end user. The static BAIT-based CDSS presents the choice recommendation without a hiding option at the top of the page. While the end user enters the values for all choice attributes, the CDSS presents the recommendation. If the end user modifies the values, the CDSS dynamically changes the recommendation in real-time. The clinical end user emphasized the importance of hiding the CDSS's recommendation when an end user enters the choice details by default. Additionally, the clinical end user suggested moving the recommendation visualization to the bottom of the page. By doing so, end user will first enter the values for the choice attributes before they interpret the recommendation of the CDSS. The foremost reason the end user provided is physicians' curiosity. Moreover, the risk exists that physicians try to achieve a 100% recommendation by slightly modifying the choice attribute values. Although this finding represents a layout issue, it is an important finding because it confirms the importance for the architecture to specify a "give me advice button".

Findings both sessions. Both Council and the clinical end user asked for additional explanation on functionalities the mock-ups present. The suggestions mainly concern the performance metrics. For example, the CDSS should clarify that the Correspondence rate does not inform how many times the CDSS was wrong but how many times a clinical end user held a minority view (see 2.2.2). Both Council and the clinical end user asked for additional explanation on functionalities the mock-ups present. The suggestions mainly concern the presentation of the performance metrics. For instance, the CDSS should clarify that the Correspondence rate does not inform how many times the CDSS was wrong but how many times a clinical end user held a minority view (see 2.2.2). This finding confirms the importance of adherence to architecture requirement QAR5 (see section 3.2.3). To clarify to CDSS developers using the architecture that it is important that

the CDSS makes the performance metrics understandable for end users, an architecture refinement ensured the architecture includes a data object containing a set of metric explanation texts. The CDSS should provide these explanation texts to end users or Councilyl as soon as the CDSS presents the performance metrics.

6.4 Summary chapter 6

The goal of chapter 6 is to evaluate the final architecture solution that chapter 5 presents. The evaluation establishes whether the formulated architecture requirements together guide the design of a valuable architecture for Councilyl. The evaluation consists of three parts: a static-oriented evaluation and two dynamic-oriented evaluations.

The static-oriented evaluation assessed the usefulness and understandability of the architecture to Councilyl. Static refers to the aspects of the architecture that one can observe without the implementation of the CDSS that the architecture describes. In essence, this is the architecture as a blueprint that CDSS developers use. Councilyl declared that the architecture forms a useful and understandable tool for the development of dynamic BAIT-based CDSSs. However, Councilyl would like to have additional supporting tools like a legend of all components and a high overview of the main architecture components. These tools store the knowledge about the architecture in the long term and support new developers, who will join Councilyl in the future and were not part of the architecture design process.

The dynamic-oriented evaluation assessed the architecture components that are observable during the run time of the CDSS. The aim of the dynamic-oriented evaluation is to show the architecture components that describe the desired functionalities in a feasible way and to highlight potential pitfalls. The dynamic-oriented evaluation consisted of the evaluation of a proof-of-technology and a series of mock-ups. The evaluation of the proof-of-technology proved that the architecture indeed describes the components that generate the updating functionalities Councilyl desires in a feasible way. However, this evaluation identified one risk: the manipulated data split in the architecture was incorrect for the type of choice data that the k-fold validation consumes. The mock-up evaluation confirms that Councilyl and a clinical end user believe that the architecture covers all essential functionalities of a dynamic BAIT-based CDSS. However, the evaluation pointed at three required architecture adjustments: guarding the privacy of choices and decision-making behaviour information within a healthcare context, additional guidance on tools and numbers the CDSS displays, and an additional feature to distinguish choices.

After processing the identified architecture adjustments, Councilyl declared that the architecture effectively guides the development of dynamic BAIT-based CDSS's. As a result, this chapter provides a positive answer to the fifth sub-question:

<p>5. To what extent does the designed architecture form an effective tool for guiding the development of a dynamic BAIT-based CDSS?</p>

Chapter 7

Reflection

This chapter presents the results of the ongoing reflection. It draws onto the ADR principle of guided emergence that prescribes that the researcher should reflect on all aspects of the design process and shed light upon seemingly incongruent perspectives (ADR principle 6) (Sein et al., 2011). The ongoing reflection resulted in a set of lessons learned. Section 7.1 presents these lessons learned. This section explains how each lesson learned informed and guided the architecture design. Section 7.2 gives a summary of the lessons learned. By doing so, this chapter presents the answer to sub-question six:

6. Considering the requirements and the evaluation, what are the lessons about how to design architectures of a dynamic BAIT-based CDSS?

7.1 The lessons learned: Reflection on the design project

The reflection took place throughout the research on an ongoing basis. The reflection concerns four focus areas: the problem framing, the emerging artefact, the theories that informed the design, the design process. Throughout the research, the lessons were noted in a logbook and used to refine the requirements and the architecture design. For each lesson learned, subsection 7.1.1 to subsection 7.1.4 explain how the lesson shaped the architecture requirements. Moreover, these sections describe what the architecture would have looked like without the requirements shaped by the iterative reflection process. By doing so, this indicates the relevance of the set of architecture requirements.

7.1.1 Problem framing

The problem formulation focuses on framing and conceptualizing a research opportunity (Keijzer-Broers, 2016). In ADR, a problem perceived in practice commonly triggers this formulation. The problem formulation functions as a ground for all remaining design activities. However, it is common for ADR studies that research findings will generate novel insights along the design process. As a result, the problem formulation set at the start of the research will evolve as the study progresses (Sein et al., 2011). This section summarizes the reflection on the problem framing and how it evolved along with the research.

Problem framing lesson 1: An architecture of a dynamic BAIT-based CDSS concerns a dynamic CDSS that changes over time as well as clinical end users who tune their decision-making behaviour over time.

At the start of this research, the design focused on an architecture that ensures a Clinical Decision Support System (CDSS) evolves by mimicking the changes in the decision-making context over time. However, research findings collected during the research process indicated that not only the CDSS tunes its internal processing to new information provided by clinical end users over time. Instead, also clinical end users adjust their internal processing over time. The clinical end user receives information from the CDSS that shapes the end user's behaviour or thinking. Internal processing refers to the processing of information that is relevant for the decision-making drives the choice recommendation of a CDSS or the choice of a clinical end user.

State-of-the-art CDSSs also influence their end users by providing choice recommendations to clinical end users. However, a dynamic BAIT-based CDSS provides also exposes the evolution of the decision-making behaviour of a pool of clinical end users over time and allows end users to investigate this for particular subgroups. By doing so, a dynamic BAIT-based CDSS produces more information compared to other types of CDSS's. As a result, the dynamic BAIT-based CDSS has a more substantial influence on the behaviour and thinking of clinical end users. Therefore, it is important to consider a dynamic BAIT-based CDSS as a CDSS that is also dynamic when its context does not change: by providing information to and, by doing is, tuning the clinical end user over time.

This insight has two primary consequences. First, it requires an extension of the set of architecture requirements. Instead of shaping an architecture of an isolated object, the challenge concerned designing components that together realize the complete operation of a dynamic CDSS in a dynamic context. Simply defining requirements that capture the technical mechanisms for storage, updating and monitoring results in an insufficient architecture that does not guide a CDSS developer in building components serving the clinical end user. Therefore, the set of requirements was extended with requirements that drive the Human-Computer Interaction (HCI) design in the context of a dynamic BAIT-based CDSS.

Second, the design of the architecture design involves organizational implications. The information provided by the CDSS has to be positioned and processed in the organization in line with the end user's preferences. Moreover, the insights that the CDSS provides require the allocation of actions and responsibilities. The formulation of such organizational implications is beyond the scope of the CDSS architecture design. However, Council needs to make clinical end users aware that the application of a dynamic BAIT-based CDSS will influence the clinical end users who use the CDSS. Moreover, Council should support end user in locating the ongoing inflow of information into the existing clinical workflow.

Problem framing lesson 2: Contextual change can be radical or gradual, each requiring a dynamic BAIT-based CDSS to describe different components.

Initially, the problem formulation framed the contextual change as simply being the change in knowledge central to the decision-making in the healthcare context. CDSS literature and interviews showed that when speaking of contextual change in the context of a BAIT-based CDSS, a distinction between two types of change exists:

- Gradual change: Gradual change occurs in small stages over a long time, rather than suddenly. An example in the context of a BAIT-based CDSS: the age attribute that becomes slightly less important over a long time.
- Radical change: Radical change occurs relatively fast and modifies the essence of clinical structures or organizational practices. An example in the context of a BAIT-based CDSS: the introduction of a new treatment that instantly affects the decision-making considerations.

At first, the research aimed at framing the problem so that it covered both types of changes. However, capturing radical change implies a significantly different challenge: incorporating radical change requires a BAIT-based CDSS to modify the structure of the choice model used for the generation of recommendations. A modification may involve adding an attribute to the model, removing an attribute from the model, or changing the attribute value range. As a result, the adaption towards radical changes in the decision-making context requires the update engine to modify and accept new model structures. In addition, the adaption requires adjusting the format in which clinical end users enter real-life choices. Finally, the choices stored in the choice bases will not fit the new model structure. Therefore, the new model cannot use the previously stored choices for model updates. As such, the choices stored before the transformation of the model structure will lose their value. Because this research focuses on allowing attribute parameters to slightly attune based on new choice information over time, solving the implications mentioned above requires additional research outside of this research scope.

As a result, incorporating radical change forms a new promising research direction. Although the insight into two types of changes sheds light upon the limitations of this research, it also explicated the scope of the requirements. Moreover, this insight marked possibilities proved to be worth to be further researched.

Problem framing lesson 3: The problem does not focus on updating for a better understanding of the choice task.

The initial aim of the research was to find alternatives that enable a BAIT-based CDSS to update according to changes in the environment and retain or even improve its Correspondence rate (accuracy) over time. This problem formulation is closely related to better informing a BAIT-based CDSS on the factual knowledge in the decision-making context. However, a conceptual distinction between these two forms is important to make. The paragraphs below substantiate why making this distinction is important.

In a static environment, incorporating new choices should enable the CDSS to understand the decision-making context better. Zikos and DeLellis (2018) even argue that the feedback loop during which the acquisition of novel information improves the understanding of a CDSS is one of the most important characteristics of CDSSs. In a static decision-making context, a CDSS supports clinical end users with choice tasks that do not change over time. However, a dynamic CDSS is still of value in a static context because the dynamic variant will understand the specific choice task better over time. By doing so, it can give better-informed choice recommendations over time. Although the knowledge fundamental to the choice will not change significantly, the choice model parameters may still be tuned in line with new experiences since the CDSS has more information. As a result, a dynamic CDSS can become more precise in a static context over time.

In a dynamic decision-making context, a dynamic BAIT-based CDSS should not only aim to understand the decision-making context better. Instead, a dynamic BAIT-based CDSS should track clinical end users' knowledge about a choice task. This knowledge will change over time. For instance, the knowledge develops or clinical end users with specific knowledge join or leave the team. In this situation, a CDSS update does better than the previous update if the CDSS is flexible enough to follow the knowledge developments in the decision-making context.

This insight drives the selection of the metrics that the architecture specifies for measuring the success of a CDSS. Instead of assessing whether the CDSS perfectly understands the decision-making context, the architecture must describe a validation process determining whether the CDSS is flexible enough to capture changing perspectives. Therefore, the architecture requirements were refined so that they force a CDSS validation to only include real-life choices that are accurate at the moment of the performance assessment. By doing so, the validation indicates the performance of the CDSS for the present situation.

Problem framing lesson 4: A dynamic CDSS that incorporates objective clinical outcomes in updates is of great value to physicians but complex to realize.

The initial problem formulation considered choice information as the set of experiment choices, real-life choices and objective clinical outcomes (OCO's). An OCO represents the result of a decision and can therefore either confirm or reject a decision. Accordingly, an OCO has the power to evaluate a clinical end user's decision-making strategy.

However, designing a CDSS architecture that includes OCO's in updates of a CDSS is complex. First of all, an OCO cannot easily be assigned a label because it will always be unclear what would have happened if the clinical end user made another choice. Second, it may take a long time before the outcome of a choice becomes clear. Finally, the specification of an outcome is subjective: what one clinical end user considers as a good outcome may another end user perceive as a bad outcome. Therefore, there does not exist a consistent classification for OCO's.

The findings of this research do prove the value of addressing the inclusion of OCO's. The interviewed clinical end users showed great interest in the inclusion of OCO's by a dynamic BAIT-based CDSS. Primarily because the inclusion of OCO's enables the CDSS to mimic how clinical end users' decision-making evolves - by doing, gaining experience, and adjusting the decision-making strategy accordingly. Second, interviewed end users declared that existing processes do not cover the storage of the choice results well. A dynamic BAIT-based CDSS could enable clinical end users to formulate, store, and organize the valuable choice outcomes. Section 9.3 presents the specific recommendations for further research on the inclusion of OCO's in updates of a dynamic BAIT-based CDSS.

7.1.2 Emerging Artefact

As the ADR framework suggests, the final architecture design results from continuous instantiating and repeatedly testing through interventions of Council (Sein et al., 2011). By doing so, the design

incorporates the insights gained along the process. The next sections list the main insights that inspired the architecture design.

Emerging artefact lesson 1: It is not problematic for a dynamic BAIT-based CDSS to incorporate real-life choices during an update that the CDSS might have assisted.

The architecture describes a CDSS that updates with information that real-life choices capture. A CDSS's recommendation might have inspired these choices. Accordingly, one may argue that an update of a recommendation generator model processes information that a CDSS inspired and, as a result, receives confirmations about its own recommendations. Consequently, the Correspondence rate (accuracy) might give a biased representation. However, this argumentation is not valid in the context of a dynamic BAIT-based CDSS. A BAIT-based CDSS codifies and captures the expert knowledge of a group of clinical end users. Accordingly, the CDSS presents the expected distribution of opinions on a particular choice task based on the expertise of these end users (Ten Broeke et al., 2021). Therefore, consulting a BAIT-based CDSS equals consulting a group of colleague clinical end users. Clinical end users also influence each other and involve in group thinking to obtain a thought-out decision in real life. Therefore, the same processes would be present in the situation without a BAIT-based CDSS.

Unlike a CDSS, the consultation of clinical end users in real-life does provide an opportunity for the back-and-forth conversations during which any notions of uncertainty or confidence can be shared (Gaube et al., 2021). However, unlike other CDSSs, a dynamic BAIT-based CDSS presents a distribution of the clinical end users' viewpoints and shows the internal agreement among the clinical end users. By doing so, it indicates the level of confidence about the recommendation. This insight in the level of confidence allows clinical end users to interpret the majority's opinion sensibly. By doing so, a BAIT-based CDSS avoids overreliance on the choice recommendation for the cases where a clinical end user was already in doubt (Gaube et al., 2021). Therefore, updating with real-life choices that a BAIT-based CDSS potentially influenced is not problematic.

However, there will always be anxious end users. Council can recommend putting more emphasis on the experiment choices in the model update to serve these users. Experiment choices stem from a controlled choice experiment without the support of a CDSS. Modifying the weight of specific choices requires the implementation of one of the Information specification Extensions (see subsection 4.3.1). Assigning experiment choices with a heavier weight may give the clinical end user confidence that the model is kept sharp with pure human knowledge over time.

Emerging artefact lesson 2: The estimation of the choice recommendation generator model does not need to include econometric techniques to combine choice information that comes from different sources.

The architecture combines experiment choices and real-life choices. Both choice types originate from a different context in which different factors shape a clinical end user's choice (M. Ben-Akiva & Morikawa, 1990). From an econometric perspective, combining the two choice types requires normalizing the variance in the unobserved factors (Train, 2009). To do so, DCM analysts commonly use an experiment to real-life choice scale parameter and multiply the attribute weights that result from either the experiment choices or the real-life choices by with this parameter (Swait & Louviere, 1993; Train, 2009; Axsen et al., 2009; Helveston et al., 2018).

The experiment to real-life choice scale parameter shows to what extent the two choice types have significantly different variances for the unobserved choice attributes (Lavasani et al., 2017). If the variance in the experimental context is relatively large, this means that the choice experiment was biased. This significant variance can also indicate that the experiment was too theoretical for a valid representation of the real-life context. If the variance in the real-life choices is relatively large, other attributes than the observed attributes may drive end user's decision-making. Consequently, Council should ascertain which factors influence end users' choices and consider refining the model structure with the right choice attributes. Alternatively, the large variance can stem from clinical end users who deviate from the desired decision-making strategy. If so, end users may need to be tuned towards the desired decision-making strategy (this poses organizational implications as meant in section 7.1.1).

Despite this common DCM practice, the architecture does not describe a scale parameter that shapes the estimation process of model updates. Three reasons further substantiate this.

- First of all, the differences in scales of the choice types are yet unknown. Moreover, it is unclear whether these differences are problematic in healthcare decision-making contexts. Before tuning the CDSS with the scale parameter, experience with the scale parameter in

different contexts is necessary. Experience in the scale differences between choice types in different context allows Councilyl to learn whether the combination of the choice types is econometrically problematic. Accordingly, Councilyl can improve the service in line with this knowledge. Therefore, the architecture does ensure each CDSS estimates a scale parameter during an update and makes the parameter available for Councilyl. By doing so, Councilyl can track the development and significance of estimated scale parameters in different contexts over time.

- Second, the scale parameter makes the decision support service complex and causes components and outcomes to be unexplainable to end users without statistical knowledge. This conflicts with the requirements formulated chapter 3.
- Finally, acknowledging the differences of the two contexts gives rise to a complex discussion with the end user. A BAIT-based CDSS codifies the knowledge of clinical end users and directly represents the decision-making of clinical end users in a particular context. When a CDSS emphasizes a potentially significant difference in how physicians make choices in an experimental context and a real-life context, it becomes vague whose recommendation an end user consults: does the recommendation stem from clinical end users in an experimental context or a real-life decision-making context?

To conclude, insight into the scale differences between the choice types allows gaining experience in the extent to which differences are present and are problematic in different contexts. Accordingly, Councilyl can determine whether the dynamic BAIT-based CDSS needs additional measures to mitigate the potential scale differences. Because the effects are unspecified, the end user should not be bothered by complex insights into statistical differences. Therefore, the architecture requirements drive a CDSS to provide Councilyl with insight into potential scale differences and ensure the scale difference to not influence the updating process.

Emerging artefact lesson 3: A dynamic-BAIT-based CDSS architecture should ensure that a CDSS eliminates choice recommendation generator model updates that stem from undesired subjectivity.

Clinical end users operating in different contexts have varying opinions about which choices are relevant for the update. Therefore, the architecture design has an adaptable structure (see chapter 5). At the same time, an adaptable approach opens the floor to subjectivity. Councilyl finds this subjectivity not problematic as long as the subjectivity is well-thought-out and the resulting model suits the clinical end users' preferences and characteristics of the decision-making context.

However, a subjective CDSS is problematic in the following two scenarios. First, if the updating does not result in an objective reflection of the real-life context within the subjectively chosen boundaries. Therefore, the architecture should maximize the objectivity of the updating processes. To this end, the architecture should ensure the processes are reliable for their purpose. Moreover, the architecture should eliminate the personal influence of individual clinical end users on the updating process. Second, if the subjectivity is random and clinical end users are unaware of the consequences of the updating preferences they specify in the user settings. Three architectural implications guard CDSS end users against encountering these situations.

1. The architecture frames the Information specification Extensions (subsection 4.3.1) as interface building blocks that only Councilyl can implement. When end users need to involve Councilyl in significant changes of the CDSS set-up, Councilyl can make end users aware of the consequences of these changes and guide finding the suiting settings.
2. When Councilyl implements an Information specification Extension, the variable settings of the specific extension still allow for subjectivity. For instance, if the implemented extension allows end users to vary the weights of all experiment choices relative to the real-life choices, clinical end user can still determine the influence of choice types on updates of a CDSS. Therefore, the architecture describes a try-out environment that allows end users to get familiar with the effects of particular settings. This environment enables end users to inspect different combinations of choice types and choice weights without modifying the updating process or recommendation generator model.
3. The architecture eliminates personal influence on the updating process. As a result, individual end users cannot determine to what extent his or her choice influences model updates. By doing so, the architecture mitigates the risk that the CDSS incorporates the emotional

status of a clinical end user. For instance, when a clinical end user is in a hurry and wants to enter the choice quickly, he or she might cut corners on specifying choice details and enters a random weight for the choice. Moreover, clinical end users in the same context may not share a similar view on when a choice should have more or less influence on the CDSS update. As such, personal influence would lead to an inconsistent weighting of choices. The inconsistent weighting makes updates of a CDSS unreliable.

Over time, Council should evaluate whether the subjectivity in the architecture requires additional restrictions in particular contexts.

Emerging artefact lesson 4: The architecture design involves privacy issues at three levels.

The architecture design deals with information accessibility issues on three levels.

1. Level 1: Between Council and the healthcare context: when the CDSS shares information on the decision-making behaviour of clinical end user's in a particular context or the performance validation of this model with Council. End users may want to hide this decision-making behaviour information for external parties.
2. Level 2: Within the healthcare context: when the CDSS shares decision-making behaviour information with the clinical end users active in the healthcare context. End users may want to hide this decision-making behaviour information for other end users in the context.
3. Level 3: Between individuals within a decision-making context: when the CDSS shares individual choices of clinical end users with the pool of clinical end users active healthcare context.

In essence, information should not be retrievable to a person. Therefore, the architecture avoids information sharing as level 3 defines. However, two additional privacy issues exist related to levels 1 and 2 in the context of a BAIT-based CDSS. First, the architecture allows clinical end users to request model updates based on a particular set of choices - for instance, all choices made by the senior physicians in the decision-making context. As a result, the risk arises that an end user can request a subgroup that only consists of one individual. Besides, clinical end users may find it inappropriate if end users can request insight into the decision-making behaviour of slightly larger subgroups. Second, the architecture should control the information that the CDSS shares with Council.

Therefore, protection measures solving these two issues are necessary. These measures must ensure the CDSS shares information in line with privacy preferences in the healthcare context. The Information Processing Agreement (IPA) that Council already has in place forms a solution for mapping the information sharing preferences of end users in a particular context. Council uses the IPA to specify what information the CDSS can share and with whom. Therefore, the IPA avoids potential privacy issues related to the decision support. When being attached to the architecture design, the IPA can guide CDSS developers in implementing information flows described by the architecture in line with the end user's information sharing preferences.

Emerging artefact lesson 5: The architecture design involves new organizational activities for Council.

The architecture describes the expansion of an existing support service. The realization of this expansion involves technological developments and organizational activities. Council needs to make decisions regarding these activities and fit these activities into the existing workflow. The activities are present in two phases:

- Pre-implementation.
 - Council should decide who the owner of the architecture is. This ownership involves keeping the knowledge captured by the architecture and additional materials that enhance the understandability of the architecture at the right place. By doing so, the right person(s) can use it as intended. In addition, the owner is concerned with the maintenance of the architecture. An adaptable architecture design requires an owner who keeps track of and processes changing preferences of end users and other factors related to components of the architecture.

- The final implementation of the CDSS depends on the three aspects. First, the specification of user settings. Councilyl has to meet with the end user and go through the user settings to find the desirable values. Second, the establishment of an Information Processing Agreement (IPA). Councilyl will need to draft a document in which Councilyl and the end users agree on the information shared with Councilyl. The shared information can either be nothing, all data (including decision-making behaviour information (relative importance) as well as model fit and performance data), or only anonymous data (model fit and performance data). The coding to implement the lines in the architecture that defines the presented information to Councilyl should follow the IPA. Third, the activities to make clinical end users aware of the need to position and deal with the information generated by the CDSS (see section 7.1.1). Councilyl needs to walk through the information the CDSS provides and discuss with the clinical end users what the value of the information is and what implications may result from having access to that information.
- Post-implementation.
 - The dynamic support service includes new features that need additional support activities from Councilyl. Moreover, different from the static service, the dynamic service runs on an ongoing basis. By doing so, the CDSS generates a continuous flow of new insights and findings that clinical end users need to understand and interpret correctly. Most likely, clinical end users will need assistance in doing so.
 - An implemented dynamic CDSS will also generate information for Councilyl. For instance, information about the performance and fit of the model that the CDSS operates. Therefore, Councilyl should assign someone to take care of this information and ensure that the necessary measures to deal with particular insights provided by this information are in place.

Emerging artefact lesson 6: The architecture of a dynamic BAIT-based CDSS can be technology-independent.

Because software technologies develop quickly and many novel innovations arise over time, the architecture will retain its value if it does not specify technology requisites and only describes components that are independent in terms of the technology they are consumed on. Another advantage of the components' independence from underlying technology is that developers can develop and change the components in the way they find most cost-effective and timely. During the design process, it turned out that an architecture can guide the development of a dynamic BAIT-based CDSS without specifying any system software and hardware components requisites at the technology layer. The two foremost reasons are as follows.

First, the architecture forms an expansion of the static BAIT-based CDSS. All insights the architecture should provide to a CDSS developer of Councilyl relate to the business and application (including software processes and information objects) level. The features and functions that the expansion needs to offer can be realized at the application level. Therefore, the expansion to a dynamic version does not generate additional requirements on the technology environment. As a result, a dynamic BAIT-based CDSS architecture design is not concerned with issues at the technology layer.

Second, a BAIT-based CDSS is a relatively isolated system because the CDSS does not rely on the functioning of other systems. The production of the data the CDSS consumes, the processes, and presents are all captured locally by the CDSS itself. It consumes choices that are entered into the CDSS by clinical end users. Therefore, the architecture design is not concerned with the technology infrastructure and formatting of communication and data sharing standards.

Emerging artefact lesson 7: CDSS-related tasks that do not require expert knowledge and skill can also form a threat to clinical end user's professional autonomy.

Existing work on CDSS design and adoption emphasizes the perceived threat of CDSSs to professional autonomy (Esmaeilzadeh et al., 2015; Khairat, Marc, Crosby, & Al Sanousi, 2018; Liberati et al., 2017; Wang et al., 2021). As such, CDSSs should ensure clinical end users feel they control the CDSS. However, these studies explain the threat as resulting from the CDSS giving recommendations and capturing the same exclusive knowledge as the clinical end user. This research involves the design of updating components that keep the CDSS accurate over time. These components do not directly challenge the knowledge and skill of clinical end users. Therefore, the research did

not treat the threat to autonomy as a dominant issue when designing in the context of a dynamic BAIT-based CDSS

However, clinical end users also prefer to control processes that do not require expert knowledge and skill. Before clinical end users feel comfortable with a CDSS that manages its updating processes, they want to become familiar with the CDSS's operation and develop trust in the CDSS. Therefore, the architecture should allow clinical end users to control the CDSS when designing for functionalities that do not directly replace the need for clinical end users' knowledge and skill. To conclude, a dynamic BAIT-based CDSS architecture should also design for the CDSS's receptiveness to control commands from clinical end users.

Emerging artefact lesson 8: Embedding the CDSS quality of transparency requires design efforts at the level of the architecture.

The BAIT approach enables a CDSS to make visible how the CDSS generates choice recommendations. Because CDSS literature stresses the need for CDSS's that explain how the CDSS infers its recommendations, the architecture initially specified an additional function that shows the explanation of the generated recommendation to the clinical end user (see ??). However, by doing so, valuable information is presented to a clinical end user when the end user is occupied with a complex choice task. Therefore, clinical end users desire a CDSS with a completely transparent structure. If the CDSS operates completely transparent, the end user can observe every part of the CDSS's operation at any time. For the architecture to drive the development of a transparent CDSS, the architecture designer should make design decisions regarding transparency that pertain to all components in the architecture. To this end, the value of transparency is handled by a Quality Attribute Requirement (QAR) instead of a Development Guiding Requirement (DGR). A DGR defines the functionalities an implemented CDSS should possess, while QARs concern aspects of the CDSS that a single component cannot capture.

7.1.3 Fundamental theories

ADR principle 2 states that theory must inform an ensemble artefact created and evaluated. Because of the novelty of the artefact, it was unclear what theories would be relevant for the design at the start of the research. Therefore, the focus was on overarching knowledge fields rather than on singular theories. This research combines theories and practices from three areas: CDSS architecture design, Discrete Choice Modeling (DCM), and Machine Learning (ML). ADR principle 6 requires continuous reflection on the theories ingrained in the architecture to identify contributions to the knowledge fields. This section presents the main lessons on applying the three knowledge fields discovered during the design process.

Fundamental theories lesson 1: CDSS architecture design literature provides useful examples for the design of a BAIT-based CDSS architecture, but the examples need modification.

Research on CDSS architecture design proposes a commonly accepted structure for CDSS architectures. This structure inspired the selection of the main components for the dynamic BAIT-based CDSS architecture design. By doing so, this research shows how the design of an architecture of a BAIT-based CDSS can reuse the main architecture components. The paragraphs below summarize the main insights gained while transforming existing CDSS components into components applicable in the context of a dynamic BAIT-based CDSS.

First of all, the architecture components need a preference-based design. CDSS architecture studies commonly propose uniform updating solutions that have a predetermined level of updating automation (see section 2.1.1). By doing so, the studies ignore potential variations in preferences on these processes. However, healthcare contexts vary. Consequently, clinical end users from different contexts have diverging preferences regarding the influence of choice types and different views on the level of updating automation (Aron, Dutta, Janakiraman, & Pathak, 2011; Eapen, 2021; Khairat et al., 2018; Pirnejad et al., 2019; Wang et al., 2021). Moreover, early involvement of clinical end-users in the CDSS development and the realization of all clinical user's needs before the development of a CDSS both increase the likelihood of CDSS acceptance (Khairat et al., 2018).

Second, when designing an architecture of a dynamic BAIT-based CDSS the designer can engage in more detailed design decisions about transparency than when designing an architecture of a traditional CDSSs. The focus of CDSS architecture design is mainly on embedding values like reliability, availability, and security. Existing CDSS designers acknowledge the value of transparency,

but evidence of examples showing how to design for transparency is limited. A plausible reason is that the more significant part of the CDSSs is opaque by nature. This opaque nature restricts the space to design for transparency. Therefore, the components proposed by CDSS design studies could not guide the design of a transparent CDSS.

Third, a dynamic BAIT-based CDSS architecture deals with new privacy issues that are not relevant for the design of traditional CDSS architectures. While accentuating transparency in the architecture CDSS design, the number of privacy concerns with which the architecture design is challenged increases. A dynamic BAIT-based CDSS stores real-life choices made by individual clinical end users, and produces information on how clinical end user's decision-making behaviour evolves over time. Consequently, a BAIT-based CDSS architecture deals with additional information flows that require the protection of personally retrievable information compared to architectures of conventional CDSSs.

Fourth, the designer of a dynamic BAIT-based CDSS architecture can ignore particular design issues. CDSS architecture design studies focus on dealing with data standards because data used by the system often comes from different sources. A dynamic BAIT-based CDSS only uses data that is locally entered and stored in a predefined format. Therefore, solutions to deal with information from other locations or systems with different standards are not relevant for the design of a dynamic BAIT-based CDSS.

Finally, architecture and CDSS architecture design literature barely specify design considerations related to the architecture as a product. The literature instead focuses on the CDSS of which the architecture describes the structure. Therefore, additional research and design efforts were necessary to design an architecture that Council can use as a tool. This research shows how a CDSS architecture can be designed as an artefact on itself, outlined in terms of requirements on the architecture level.

Fundamental theories lesson 2: Discrete Choice Modeling (DCM) offers promising features to a dynamic CDSS. However, not all DCM theories and practices are applicable in the context of a CDSS architecture because of the diverging goals of DCM and decision support.

A BAIT-based CDSS codifies decision-making knowledge using Discrete Choice Modeling (DCM). (Ten Broeke et al., 2021) already prove that the use of DCM provides a BAIT-based CDSS with characteristics that have great potential for the support of clinical choice tasks. This research shows that the DCM has additional values for decision support in ever-changing healthcare contexts. The list presented below gives the three most prevalent values:

1. DCM enables the development of transparent CDSSs that make visible how the decision-making behaviour in a healthcare context evolves. Most of the traditional CDSS's only provide clinical end users with choice recommendations.
2. DCM enables a dynamic BAIT-based CDSS to easily collect novel choice information that the CDSS for updating the recommendation generator model over time. Instead of combining different data sources, a dynamic BAIT-based CDSS can collect this choice information locally (see subsection 2.2.1). Moreover, the information always has a similar format. As such, the components of dynamic BAT-based CDSS do not depend on other systems. This independence eliminates the need for complex solutions to prepare and combine historic information originating from different sources.
3. DCM provides a dynamic BAIT-based CDSS with two characteristics that mitigate the potential issue resulting from updating a CDSS with choices that the CDSS might have assisted (see section 7.1.2).

Despite the advantages of DCM for CDSS design, an architecture designer should not blindly copy DCM theories, practices, and characteristics. The goal with which DCM studies generate theoretical and practical suggestions significantly differs from the goal fostered with a decision support service. The goal of DCM is to find the parameters that best describe the data generating process (DGP) to convey the estimated model as the true representation of the data from which behavioral inferences can be made (Van Cranenburgh, Wang, Vij, Pereira, & Walker, 2021). A BAIT-based CDSS aims to utilize DCM to generate insights that maximize the quality of the decision support. Because of these diverging intentions, the final set of requirements does not drive the architecture designer to include theories and practices suggested in DCM literature to

better approach the DGP. The list below gives two examples of how the architecture was affected by this lesson learned:

1. The architecture does not include the scale parameter that DCM literature proposes for a joint choice model estimation (for an explanation of the scale parameter, see section 7.1.2). The inclusion of the parameter resulted in a choice recommendation model that is justifiable from an econometric perspective. However, the model erased the noise present in the real-life context, like time pressure affecting the decision. Interviewed clinical end users interviewed emphasized that they prefer the model to include this real-life noise because it makes the model realistic. Moreover, the scale parameter makes the updating process more complex. As a result, end users will find it harder to understand and track the CDSS behaviour. This complexity conflicts with the architecture requirements on transparency (QAR5, QAR6). Therefore, requirements forcing the architecture to correct the model update with a scale parameter were excluded.
2. The architecture does not specify components that check for the significance of the estimated choice model parameters. DCM studies aim at significant outcomes that allow for generalization to a population. However, in the context of decision support, the findings only need to reflect the decision behaviour of the clinical end users within the sample. Therefore, generalization to a population is irrelevant. Accordingly, the set of requirements does not require the architecture to include components that assess the significance of values estimated during an update.

Fundamental theories lesson 3: Machine Learning techniques and theories are useful, but need modification for a proper application of the techniques and theories in the context of Discrete Choice Modeling-based decision support.

At the start of the research, it was expected that Machine Learning (ML) techniques could be directly used in the architecture design. The main reason for this assumption was that ML studies focus on building and improving models based on experience to make more accurate predictions. This focus approaches the goal of a dynamic BAIT-based CDSS. Moreover, DCM has similarities with ML: both are grounded in statistical theory, face similar challenges (for instance, unbalanced data sets, and the trade-off between model complexity and understandability), and aim to generate predictions that are replicable and flexible ((Van Cranenburgh et al., 2021)).

However, DCM and ML are significantly different. In contrast to DCM, the training of a ML model involves labelled records (Ten Broeke et al., 2021). These records function as examples for the to be modeled input-output behaviour. With the labelled records, ML researchers aim to find the model that is best capable of out-of-sample generalizing. By doing so, the resulting model approaches the correct relationships between the relevant variables (Van Cranenburgh et al., 2021). To this end, ML assumes that the data generating process (DGP) is unknown. On the contrary, a DCM researcher believes that the decision-making that the researcher analyzes follows a particular decision rule (Alwosheel, 2020). An example of a decision rule is Random Utility Maximization (RUM), which states that a decision-maker aims to maximize the utility he or she obtains from a choice alternative. As a result of these divergent approaches, ML models are black boxes while DCM models show how clinical end users make decisions (Ten Broeke et al., 2021). Another important difference is that ML starts from a deterministic ground truth, while a BAIT-based CDSS is probabilistic in nature (see section 2.2.2). Due to these differences, a designer of a dynamic BAIT-based CDSS architecture can reuse ML theories and techniques. However, the designer needs to tune them, so they are applicable for a DCM-based decision support technology. The list below shows four examples of adjustments that were necessary for the architecture for Council:

- The architecture incorporates the ML-based k-fold validation technique. However, the reuse of this technique in the context of a dynamic BAIT-based CDSS required two adjustments. The first adjustment concerns the random data split of the k-fold cross-validation. The adjustment ensures that the CDSS equally distributes similar choice tasks over the training and validation set. The second adjustment ensures that the validation set only includes the recent choices representing the present decision-making behaviour in the context. For a more detailed explanation of both adjustments, see subsection 4.2.5.
- The architecture involves ML-based performance metrics. However, the metrics have different names that better fit the probabilistic ground-truth basis of a dynamic BAIT-based CDSS

(see section 2.2.2). In addition, DCM model diagnostics metrics and a metric uniquely designed for a BAIT-based CDSS (the Confidence Representation rate) complemented the set of ML-based metrics.

- The architecture involves ML-based performance metrics. However, the architecture allows end users to use the outcomes differently. For ML, the goal is to find the model with the best performance. In the context of a BAIT-based CDSS, the performance of a recommendation generator model should not be the only factor driving the acceptance of the model. Beyond its performance, an end user may perceive a model as desired or undesired given the accepted norms and values in the context.

7.1.4 Design Process

Finally, the research generated lessons that inspired the organization of the design process. The design process involves activities key for designing an architecture in a situated context and formalizing the learning in this context. Although the research started with a research plan, findings discovered along the process required revising the initial schedule. Moreover, some practices were not part of the initial research planning but turned out to work well for achieving the research goal. The sections below present the main lessons learned regarding the design process.

Process lesson 1: Designing an architecture as an artefact involves design considerations at two levels.

Although the artefact subject to design is an architecture, the design process was highly correlated with specifications at the level of the CDSS. Therefore, designing the architecture involves design considerations at two levels: at the level of the architecture and the level of the CDSS that the architecture describes. However, architecture and CDSS architecture design literature does not explicitly support designing an architecture that functions as an artefact for CDSS providers. Instead, current work presents the architecture as a tool to communicate and illustrate the features of the CDSS. The difference is that in case of the latter, the architecture design is not explicitly bound to architecture requirements. Instead, the architecture designer focuses on the requirements of the CDSS because the designer uses the architecture to find the right CDSS design.

In addition, because designing an architecture involves design considerations at two levels, it significantly differs from designing other kinds of IT artefacts. The design of these other IT artefacts only requires design choices at a single level. As a result, no studies were in place that could function as an example for designing the architecture for Council. To deal with the design considerations at the two levels, the research process included the following two additional steps:

- The requirement identification involved both representatives of Council and clinical end users. Council provided insight into the requirements that the architecture must satisfy to be a useful and understandable artefact for Council. The clinical end users marked the features a CDSS must possess. To translate the information collected at different levels into information that directly informs the architecture design, the requirement formulation followed three categories: Client Architecture Requirements Quality Attribute Requirements, and Development Guiding Requirements.
- The evaluation involved an assessment at the architecture level and CDSS level. First, the evaluation assessed whether Council finds the architecture is useful and understandable. Next, the evaluation assessed whether the architecture specifies the correct CDSS functionalities. To this end, the evaluation included a dynamic-oriented assessment of all components in the architecture that are only visible during CDSS run time. During this dynamic-oriented evaluation, Council and clinical end users judged the relevance of the functionalities described in the architecture. Council and the clinical end users functionalities could evaluate the functionalities via a proof-of-technology and a series of mock-ups.

Process lesson 2: The ADR design cycles should not follow the extensions of the adaptable architecture design.

The ADR framework proposes repeated Building, Implementation and Evaluation (BIE) cycles (Sein et al., 2011). The initial planning assigned each cycle to a specific Information specification Extension and update activation extension (for an explanation of these extensions, see subsection 4.3.1 and subsection 4.2.1). However, structuring the design process using each cycle for a different part of the architecture is not recommendable. Two reasons further substantiate this:

1. At the start of the design process, it is unclear which extensions the optimal design requires. Commonly, findings done along an ADR design process iteratively shape the problem formulation and the emerging artefact (Sein et al., 2011). Therefore, a design process cannot be planned based on extensions or other parts that a designer expects to need for the architecture design.
2. Each extension or part of the design needs several design iterations. When using a single BIE cycle per extension, each extension only receives attention during one BIE cycle.

Based on the experience gained during this research, dynamic BAIT-based architecture designers are recommended to already design each extension conceptually during the first BIE cycle. Each successive cycle can further shape or remove all parts of the design.

Process lesson 3: The architecture design requires technical knowledge and experience.

The design of a dynamic BAIT-based CDSS architecture is associated with multiple technical challenges. The interviewed clinical end users had little understanding of the technical terms and found it hard to make explicit what technical aspects they need or preferred. The end users declared to fully trust the design team for finding the technical design solutions. Therefore, the designer should find suitable technical solutions for a dynamic BAIT-based CDSS architecture individually. To this end, the designer needs to possess knowledge about DCM and statistics beyond DCM. Additionally, the designer should incorporate scientific knowledge on the technological concept(s) for all architecture components to ensure the architecture describes statistically correct processes. Finally, a designer is recommended to conduct a dynamic-oriented evaluation of the architecture that assesses the effectiveness of all technical components.

Process lesson 4: The creation of mock-ups enhances the architecture design.

Although the design process focuses on the design of the architecture, the development of CDSS mock-ups formed an effective manner to evaluate and improve the architecture design. Three reasons explain why the mock-up development positively influenced the architecture design. First, mock-ups stimulate collaboration with stakeholders. Visualizations of components described in the architecture make it easier to ascertain if the designer and stakeholders are on the same page. Second, mock-ups help the designer to think about design aspects that the designer initially overlooked. During this research, the creation of the mock-ups revealed that particular information flows to interfaces were missing in the architecture. Finally, mock-ups support the evaluation of the architecture with stakeholders at the level of the CDSS. By doing so, mock-ups allow assessing whether an architecture defines the right features and functions.

7.2 Summary chapter 7

The goal of chapter 7 is to present the results of the ongoing reflection of this design research. The reflection enabled making the step from designing for the particular problem instance of Council to applying that learning to a broader class of problems. The ongoing reflection resulted in a set of lessons learned. These lessons shaped the problem framing, the emerging artefact, the theory ingrained in the artefact, and the design process.

The lessons learned regarding the problem framing show that in the context of a dynamic BAIT-based CDSS, both a CDSS and a clinical end user tune their internal processing over time. Moreover, the lessons indicate that contextual change is either radical or gradual and that the problem formulation of this research relates to gradual change. Finally, the reflection emphasized that the problem formulation does not require a solution that enables a CDSS to better understand a choice task over time but rather to become flexible enough so that the CDSS can generate accurate decision support in an ever-changing context. The lessons learned on the emerging artefact concern both technical and social design aspects. Four lessons concern technical design aspects.

1. It is not problematic for a dynamic BAIT-based CDSS to update with choices the CDSS has assisted because the characteristics of a dynamic BAIT-based CDSS mitigate the effects that might result from updating with these choices.
2. The architecture should not include econometric techniques to deal with scale differences between information that experiment choices and real-life choices capture. Including these

techniques would make a dynamic BAIT-based CDSS too complex to comprehend for clinical end users.

3. A dynamic-BAIT-based CDSS architecture should ensure that a CDSS eliminates choice recommendation generator model updates that stem from undesired subjectivity.
4. An architecture can guide the development of a dynamic BAIT-based CDSS without specifying software and hardware prerequisites.

Three lessons concern social design aspects.

1. Privacy and organizational issues play a key role when designing a dynamic BAIT-based CDSS architecture.
2. Clinical end users like to control CDSS components that do not require physicians' knowledge and skill. Therefore, the architecture should ensure that a dynamic BAIT-based CDSS is receptive to clinical end users.
3. Clinical end users want a CDSS to operate completely transparent: simply complementing choice recommendations with an explanation of how a CDSS created the recommendation is insufficient. Instead, the designer should ensure that the architecture only describes components that clinical end users understand, and that make the internal change of the CDSS tractable for clinical end users.

Theories and practices provided by Discrete Choice Modeling, Machine Learning, and CDSS architecture design are beneficial for the design of a dynamic BAIT-based CDSS architecture. However, every theory or practice requires customization before a CDSS developer can apply the theory or practice in the context of a dynamic BAIT-based CDSS. To finish, lessons learned shaped the design process guiding this research. The most important lesson concerning the process is that the architecture design involves design consideration at the level of the architecture and the level of the CDSS.

Each lesson further informed and shaped the list of architecture requirements and, by doing so, the architecture design for Council. The combination of the lessons learned answers the sixth sub-question:

6. Considering the requirements and the evaluation, what are the lessons about how to design architectures of a dynamic BAIT-based CDSS?

Chapter 8

Generalization

The goal of this chapter is to make the move towards the generalization of the research findings. This chapter draws on the ADR principle of generalized outcomes. This ADR principle claims that the learning should be abstracted to a class of field problems (ADR principle 7) (Sein et al., 2011). Chapter 7 presents the lessons learned that this research identified. This learning inspires the translation of the tested architecture requirements into design principles that guide the design process of a dynamic BAIT-based Clinical Decision Support System (CDSS) architecture outside a situated context. Section 8.1 presents the design principles resulting from this translation step. Next, section 8.2 describes the relationships between the coherent set of design principles. To illustrate the contribution of the generalized findings to the design science knowledge, section 8.3 explicates the novelty and added value of each design principle. The chapter closes with a summary in section 8.4.

8.1 Design principles

This research aims to find the design principles that guide the design of a dynamic BAIT-based CDSS architecture outside a situated context. Design principles capture knowledge gained along with the process of building a solution and encompass knowledge about creating other instances that belong to this class (Dasgupta et al., 1996). By doing so, design principles connect the generalized outcomes to a class of solutions, and a class of problems (Sein et al., 2011). As a result, design principles function as recommendations on how a designer should design artefacts that solve comparable problems. Here, it is not about the generalization from a sample to the population, but rather about transportability: the usability of the design principles in problem contexts outside the study context (Degtiar & Rose, 2021; Lesko et al., 2017). Within this research, the class of problems refers to the challenge of designing a dynamic BAIT-based CDSS architecture that functions as a guiding tool for developers of transparent and dynamic CDSSs who aim to serve various healthcare decision-making contexts. The architecture forms a solution instance that represents a class of solutions. In this research, the class of dynamic CDSS architectures: architectures of CDSSs that clinical end users apply in ever-changing decision-making contexts.

Generalizing findings of ADR research is challenging because the outcomes of an ADR study are highly situated. Therefore, this research followed a structured approach to generalize the situated outcomes. The approach consists of clustering the tested requirements into overarching themes of design principles. The choice for this approach stems from three reasons. First, the research identified 43 architecture requirements. Therefore, the number of requirements is too high for a direct translation into design principles, which indicates the need for clusters of requirements. In addition, the architecture requirements are interrelated. The interrelation between the requirements indicates that the requirements are dependent on each other and can form clusters of coherent requirements. Finally, the lessons learned (see chapter 7) demanded the definition of particular design principles. By doing so, these lessons could guide the formulation of particular requirement clusters.

Making the conceptual move to the generic design principles involves three steps. The first step is to write down all problem context-related architecture requirements on post-its. The set of post-its gives a clear overview of all requirements subject to the clustering. The second step is to cluster the requirements into overarching themes. This second step also involves consulting the lessons learned (see chapter 7) and determining what design principles these lessons stipulate. By

doing so, the lessons guide the clustering of the requirements into themes that align with important design lessons. Some requirements related to multiple themes, indicating that relationships exist between the design principles. The final step consists of using the overarching themes of clustered requirements to formulate design principles.

The generalization process resulted in a coherent set of ten design principles. Table 8.1 provides an overview of all ten design principles. Because the lessons learned inspired their formulation, the design principles are based not only on the truth - the requirements - but also on what is important for designers to do to avoid design obstacles encountered during this research. The sections below explain all design principles. Each section describes a principle and explains the internal cohesion between the underlying architecture requirements. The code between brackets in each section refers to the architecture requirements that section 3.1 introduces (CAR: Client Artefact Requirement, QAR: Quality Attribute Requirement, DGR: Development Guiding Requirement). Moreover, each section explicates how the lessons learned informed the formulation of a design principle.

Table 8.1: *Overview of the design principles for designing a dynamic BAIT-based CDSS architecture.*

Reference	Catch word	Design principle
Design principle 1	Adaptable design	The architecture should have a technology-independent, adaptable structure with extensions that enable customization of the updating automation level and the choice information that updates incorporate.
Design principle 2	Objectivity maximization within the subjective boundaries	The architecture should maximize the objectivity with which choice information is processed during an update, given the clinical end user's deliberately chosen updating preferences.
Design principle 3	Goal-based interaction	The architecture should only define interaction flows that are always available and from which a clinical end user directly or indirectly benefits.
Design principle 4	Tractability of change	The architecture should only describe CDSS components and outcomes that are comprehensible for clinical end users and that make the change in the choice recommendation generator model tractable.
Design principle 5	Receptivity to user input	The architecture should force a receptive CDSS that induces interaction with end users and give end users control over the change in the choice recommendation generator model.
Design principle 6	Differentiation in consumed choices and produced information	The architecture should distinguish types of choices that different CDSS processes consume and types of information that a CDSS produces for different receivers.
Design principle 7	Mutual learning	The architecture should force the development of a CDSS that enhances mutual learning between clinical end users.
Design principle 8	Architecture intuitiveness	The architecture should have an intuitive design for developers with knowledge and skill in Discrete Choice Modeling.
Design principle 9	Privacy of choice information and decision-making behaviour information	The architecture should not contain decision-making information flows that can be traced back to individual clinical end users or are undesired by clinical end users in the healthcare context.
Design Principle 10	Explication of the organizational activities	The architecture should be complemented with a description of the organizational activities needed from the CDSS provider and a procedure that guides the provider in executing these activities.

8.1.1 Design principle 1: Adaptable design

Design principle: The architecture should have a technology-independent, adaptable structure with extensions that enable customization of the updating automation level and the choice information that updates incorporate.

The first design principle captures two structural aspects. The first structural aspects is that the architecture should be adaptable. An adaptable architecture needs to contain three elements: core components, optional extensions, and a set of modifiable process values (CAR10). The core includes all components that are essential for a dynamic BAIT-based CDSS independent of the preferences of end users in a healthcare context. The extensions are optional components that can be implemented if desired by the end user. The extensions should allow end users to shape two aspects of the CDSS: the weight with which an update includes different types of choices and the level of updating automation. Next to the extensions, the architecture should not predetermine the

values influencing the behaviour of a CDSS's processes (DGR6). An example is the value for the majority threshold (see section 5.3). When the architecture includes these extensions and these modifiable process values, the architecture enables CDSS developers to customize the CDSS so that the CDSS reflects end users' preferences. The importance of customization results from the differences between healthcare contexts and the varying preferences regarding a dynamic BAIT-based CDSS of end users in these contexts (see Fundamental theories lesson 1 in section 7.1.3). Because of these differences, the architecture should be able to guide the development of CDSS's with different characteristics (CAR10).

The architecture should also be adaptable over time (CAR11). Because of the ever-changing nature of healthcare decision-making contexts, clinical end users' preferences will change over time. The changing preferences have two implications for the architecture design. First of all, a developed CDSS may need a modification to reflect end users' preferences later in time. Therefore, the architecture describes extensions and modifiable process values (see the paragraph mentioned above) that can always be adjusted with new extensions or different process value assignments. Second, the architecture designer should be able to expand the architecture with new components. To ensure the addition, removal, or replacement of architecture components will not have unintentional effects, the architecture should define all dependencies between the components in the architecture (CAR9). When the architecture defines all these dependencies, the architecture informs a CDSS developer about the possible effects of an architecture modification on all components.

The second structure aspect is that the architecture should have a technology-independent structure. Architecture design research commonly describes system components at three levels: the business level, application level, and technology level (for an explanation, see subsection 4.1.1). The architecture of a dynamic BAIT-based CDSS should specify business processes because a dynamic BAIT-based CDSS relies on organizational activities executed by the CDSS provider and clinical end users (see Emerging artefact lesson 5 in section 7.1.2 and Problem framing lesson 1 section 7.1.1). These activities determine the interaction that a dynamic CDSS must include to serve clinical end users properly. An architecture designer typically defines these activities as business processes at the business layer of the architecture. Therefore, an architecture of a dynamic BAIT-based CDSS includes processes at the business layer. The architecture should specify application components because the architecture should inform CDSS developers on the components that a dynamic BAIT-based CDSS needs to incorporate new choice information during an update. An architecture designer typically describes these types of components at the application layer. Therefore, an architecture of a dynamic BAIT-based CDSS includes components at the application layer.

The architecture can guide the development of a dynamic BAIT-based CDSS without specifying any system software and hardware requisites (see Emerging artefact lesson 6 in section 7.1.2 and see Fundamental theories lesson 1 in section 7.1.3). By doing so, the architecture only describes components that are independent of the technology they are consumed on and does not represent components at the technology layer. This technology-independent design is beneficial for an architecture to retain its value over time (see Emerging artefact lesson 6 in section 7.1.2). Table 8.2 gives an overview of the requirements fundamental to design principle 1.

Table 8.2: *Architecture Requirements design principle 1: Adaptable design.*

Reference	Architecture Requirement
DGR6	The architecture should force the development of a CDSS that operates a modifiable majority threshold and level of acceptance.
CAR10	The architecture should be adaptable to the preferences present in all healthcare decision-making contexts.
QAR3	The architecture should never force a fully automated CDSS.
DGR7	The architecture should force the development of a CDSS that estimates a new choice recommendation generator model according to the choice types and weight specification clinical end users selected as soon as clinical end users deem this model inaccurate or undesired for decision support.
CAR9	The architecture should inform on all dependencies between architecture components.
CAR11	The architecture should be adaptable to the potential future preferences in all healthcare decision-making context.
CAR12	The architecture should always be integrable with the service environment of the CDSS service provider.

8.1.2 Design principle 2: Objectivity maximization within the subjective boundaries

Design principle: The architecture should maximize the objectivity with which choice information is processed during an update, given the clinical end user's deliberately chosen updating preferences.

The foremost goal of a dynamic BAIT-based CDSS architecture is to define a CDSS that incorporates new choice information during an update of the recommendation generator model, so the CDSS's recommendations align with the changes in the decision-making context. Therefore, the architecture should define components that ensure the CDSS can incorporate new experiment choices and new real-life choices during ongoing use (QAR8, DGR10).

The preferences regarding the types of choices that should inform an update varies between health-care contexts. Therefore, design principle 1 - adaptable design - guides designers to develop an architecture with extensions that allow end users to customize the type of choices that inform an update (see subsection 8.1.1). However, letting end users select the choice types that will inform an update makes the update subjective (see Emerging artefact lesson 3 in section 7.1.2). Therefore, an architecture of a dynamic BAIT-based CDSS should specify two approaches that avoid the development of a CDSS that is unintentionally informed with a distorted representation of the decision-making context.

The first approach is to ensure the choices informing an update are deliberately selected. By doing so, the clinical end user is aware of the consequences of customization (see Emerging artefact lesson 3 in section 7.1.2). The awareness implies that the customization is intentional and results from a particular purpose rather than from arbitrariness. The intentional customization justifies the subjectivity.

The second approach is to maximize the objectivity with which a CDSS processes choice information during an update. The maximization of objectivity relates to two requisites (see Emerging artefact lesson 3 in section 7.1.2). The first requisite is that the architecture only describes reliable updating processes as part of the model update engine (CAR1, QAR9, DGR2). The architecture design requires extra attention on reliability because of the Machine Learning (ML) techniques that inform the design of the components in the architecture. The reliable application of ML techniques in the context of dynamic BAIT-based CDSSs is not similar to the reliable application in the context of ML-based CDSSs (see Fundamental theories lesson 3 in section 7.1.3). Therefore, an architecture designer should actively monitor the reliability of all components while designing the architecture. The second requisite is that the architecture should eliminate any personal influence of individual clinical end users on updates (QAR7). Personal influence refers the influence of a clinical end user's mental state. For instance, the influence of being in a rush or being emotionally challenged on an end user's choice. The selection of the choice information that the model update engine processes would become inconsistent if every end user can determine how much their choice should influence the model updates themselves. Each end user follows different criteria to choose a weight with which a particular choice type influences the update. Moreover, an end user may use different criteria at different points in time. Consequently, it is hard to retrieve why particular choice types informed an update.

A final notion regarding design principle 2 is necessary. The model recommendation generator model estimated during an update aims to represent the pool of clinical end users in the particular context rather than a wider group of clinical end users outside this sample (see Fundamental theories lesson 2 in section 7.1.3). Although econometricians and DCM-analysts might expect this, the design principles concerning the updating reliability do not force the estimated recommendation generator model to be statistically significant.

To conclude, the architecture should never be limited to the maximization of objectivity. Instead, the architecture should maximize objectivity within the boundaries that clinical end users set to reflect their decision support preferences. Clinical end users define which choices should shape an update, and a CDSS has the components to objectively update the choice recommendation generator model. Table 8.3 gives an overview of the requirements fundamental to design principle 2.

Table 8.3: *Architecture Requirements design principle 2: Objectivity maximization within the subjective boundaries.*

Reference	Architecture Requirement
CAR1	The architecture should define the model update engine only with processes that have proved to achieve the goal for which the architecture includes the processes.
QAR9	The architecture should only define components that work statistically correct.
QAR7	The architecture should avoid the development of a CDSS that allows clinical end users to directly determine the importance of a single real-life choice in the model estimation.
QAR8	The architecture should force the development of a CDSS that only updates the choice recommendation generator model according to contextual changes captured by the experiment choices and real-life choices the CDSS is informed about.
DGR10	The architecture should force the development of a CDSS that allows replacing the experiment choices with experiment choices from a new choice experiment.
DGR2	The architecture should force the development of a CDSS that does not interchange patient-specific data to deal with incomplete real-life choices.

8.1.3 Design principle 3: Goal-based interaction

Design principle: The architecture should only define interaction flows that are always available and from which a clinical end user directly or indirectly benefits.

A dynamic BAIT-based CDSS produces more information compared to traditional CDSSs. Therefore, a dynamic BAIT-based CDSS architecture designer will need guidance on designing for the interaction that the exchange of this information with clinical end users requires (see Problem framing lesson 1 in section 7.1.1). The interaction between a CDSS and clinical end users is associated to a contradiction that complicates the design for this interaction. On the one hand, clinical end users have stressful working days and aim to fully focus on patients' well-being. Interrupting clinical end users or keeping them occupied without a clear purpose will frustrate end users and negatively affect end users' attitudes towards a CDSS. On the other hand, clinical end users do not like to be excluded by a CDSS or unable to command a CDSS (see Emerging Artefact lesson 7 in section 7.1.2). They rather control a dynamic BAIT-based CDSS and observe the outcomes resulting from updates.

To deal with this contradiction, a designer should be careful that the architecture only describes interaction flows between a CDSS and clinical end users that ask information or provide information from which clinical end users directly or indirectly benefit. For instance, a clinical end user is encouraged to enter additional real-life choice information, like his or her confidence with regard to a choice, if this leads to additional insights into the decision-making behaviour of clinical end users and mutual learning (QAR11, DGR15, DGR16). Another example is that a clinical end user might want to be informed about the completion of an update but does not want to investigate the outcomes resulting from a model update at the same time. The interaction should therefore consist of a confirmation without the presentation of any update outcomes. By doing so, the interaction flows in the architecture will not distract clinical end users with unsolicited information.

Moreover, the architecture should ensure that clinical end users are not hindered while interacting with a CDSS (QAR10, QAR12, QAR2). By doing so, the architecture minimizes the time and effort end users need to complete particular tasks with a CDSS. A specific architecture component enables this undisturbed interaction. The architecture should force the development of CDSS components that enable clinical end users to request support on particular functionalities or outcomes that a CDSS generates (QAR13). For instance, in the form of additional explanations. The notion of 'request' is important because end users will perceive unsolicited support as superfluous. With a support component in place, a CDSS can help clinical end users facing an obstacle during the interaction with a CDSS to effectively continue their task as quickly as possible. Table 8.4 gives an overview of the requirements fundamental to design principle 3.

8.1.4 Design principle 4: Tractability of change

Design principle: The architecture should only describe CDSS components and outcomes that are comprehensible for clinical end users and that make the change in the choice recommendation generator model tractable.

The resources that assist clinical end user's decision-making are ideally fully transparent because end users' choices concern patients' well-being. Despite that a dynamic BAIT-based CDSS directly

Table 8.4: *Architecture Requirements design principle 3: Goal-based interaction.*

Reference	Architecture Requirement
QAR10	The architecture should minimize the time and number of activities that a CDSS requires from clinical end users to fulfil a clinical end user's goals with the CDSS.
QAR11	The architecture should force the development of a CDSS that only asks a clinical end user to enter choice-specific data when entering a real-life choice from which the clinical end user will benefit later in time.
QAR12	The architecture should force the development of a CDSS that clinical end users can always use for a choice recommendation request and measurement of a real-life choice.
QAR13	The architecture should force the development of a CDSS that enables clinical end users to always request online support on the CDSS components and outcomes these components produce.
DGR15	The architecture should force the development of a CDSS that allows clinical end users to request the relative importance of the choice attributes of all model updates.
QAR2	The architecture should minimize the number of intervening actions needed from Council that are not requested by a clinical end user.
DGR16	The architecture should force the development of a CDSS that allows clinical end users to request the relative importance of the choice attributes for a by the clinical end user selected subgroups.

reflects the changes in clinical end users' own expert knowledge, clinical end users are not likely to accept a dynamic CDSS if it does not make transparent how it processes these changes during an update.

The DCM ground of a dynamic BAIT-based CDSS allows a dynamic BAIT-based CDSS to make the changes in the choice recommendation generator model fully transparent (see Problem framing lesson 1 in section 7.1.1). Accordingly, an architecture designer should exploit the DCM characteristics and describe components in the architecture that ensure the visibility of the changes in a dynamic BAIT-based CDSS's recommendation generator model (see Fundamental theories lesson 1 in section 7.1.3). However, simply presenting an explanation of how a CDSS derived the recommendation when a clinical end user requests a choice recommendation is insufficient (see Emerging Artefact lesson in section 7.1.2). Instead of capturing transparency with an additional CDSS functionality, the architecture designer should treat transparency at the architecture level. By doing so, the entire CDSS structure aligns with the value of transparency. To treat transparency at the architecture level, the designer should consider two design focus areas.

The first area is the tractability of a CDSS's components and the outcomes that these components produce. The architecture designer should ensure that a CDSS provides end users with tools to track the changes in the recommendation generator model. To this end, a CDSS should inform clinical end users about the main updating activities (DGR13, DGR14). For instance, by sending a confirmation when a CDSS completes a model update. Moreover, a CDSS should inform end users about changes resulting from these updating activities. Therefore, the architecture should include components that enable a CDSS to visualize how the parameters of the recommendation generator model change (QAR6, DGR15) and how a model's performance changes over various updates (DGR11). Because clinical end users tend to forget to check the performance regularly, the architecture should describe a model quality monitor that continuously assesses a recommendation generator model's performance on accurate choice tasks (DGR4, DGR5).

The second area is the comprehensibility of a CDSS's components and the outcomes that these components produce. If end users cannot understand these components and the outcomes, the tractability of these components and outcomes is meaningless. The essence of all CDSS components should be understandable regardless of any knowledge about Discrete choice modeling, statistics, and Machine Learning (QAR5). Therefore, the architecture designer should be careful with describing complex statistical techniques and performance metrics in the architecture. For instance, the inclusion of the scale parameter made a CDSS's model update engine too complex for clinical end users to understand and track the changes resulting from model updates (see Fundamental theories lesson 2 in section 7.1.3 and Emerging artefact lesson 2 in section 7.1.2). Moreover, the architecture should describe an environment where end users can try out model updates (QAR4). This environment will help end users to become familiar with a CDSS's components and outcomes. Table 8.5 gives an overview of the requirements fundamental to design principle 4.

Table 8.5: *Architecture Requirements design principle 4: Tractability of change.*

Reference	Architecture Requirement
QAR5	The architecture should only include components and provide clinical end users with outcomes that a clinical end user without any knowledge about Discrete Choice Modelling, statistics, and Machine Learning can understand.
QAR4	The architecture should avoid the development of a CDSS that forces clinical end users to accept a choice recommendation generator model version.
DGR14	The architecture should force the development of a CDSS that confirms the completion of a choice recommendation generator model update.
DGR13	The architecture should force the development of a CDSS that presents an alert to clinical end users when the level of acceptance has been reached and the choice recommendation generator model is not updated yet.
DGR9	The architecture should force the development of a CDSS that makes the date at which a clinical end user entered a choice used for the model validation transparent.
DGR5	The architecture should force the development of a CDSS that copies real-life choices exceeding the majority threshold but deviate from a clinical end user's choice to a separate database.
DGR4	The architecture should force the development of a CDSS that compares each choice recommendation with the majority threshold and the clinical end user's choice as soon as an end user enters a real-life choice into the CDSS.
QAR6	The architecture should force the development of a CDSS that makes its internal changes transparent for clinical end users.
DGR11	The architecture should force the development of a CDSS that enables clinical end users to request the performance metrics for all choice recommendation generator model updates.
DGR15	The architecture should force the development of a CDSS that allows clinical end users to request the relative importance of the choice attributes of all choice recommendation generator model updates.

8.1.5 Design principle 5: Receptivity to user input

Design principle: The architecture should force a receptive CDSS that induces interaction with end users and give end users control over the change in the choice recommendation generator model.

The architecture should ensure that an implemented CDSS is receptive to the input of clinical end users. Receptivity to clinical end users is important for the acceptance of a CDSS. Two reasons further substantiate why this is important. The first reason is that if a CDSS is receptive to clinical end users' input, this will give end users the feeling they have control over a CDSS. Clinical end users are experts who have gained much experience over time. As a result, they have developed a sense of professional autonomy. The tasks of a dynamic BAIT-based CDSS do not directly challenge the expert knowledge and skills of clinical end users. However, these tasks still form a threat for a clinical end user's professional autonomy when a CDSS unsolicitedly controls these tasks (see Emerging Artefact lesson 7 in section 7.1.2). Being able to control the tasks of a dynamic CDSS will result in a positive attitude towards a CDSS because it leaves space for clinical end users' autonomy (Esmailzadeh et al., 2015; Siau & Shen, 2003). Second, receptivity to clinical end users' input gives end user the feeling they can influence a CDSS. By doing so, a receptive CDSS approaches real-life group thinking (see Emerging artefact lesson 1 in section 7.1.2). Group thinking allows clinical end users to exchange information about a choice task and influence each other's decision-making. Because a receptive CDSS approaches this real-life phenomenon, end users will deem a CDSS as something they are familiar. As a result, clinical end users will be more likely to develop a positive attitude towards a receptive CDSS.

To enhance the receptivity to clinical end users, two manners that both stem from the DCM ground of the BAIT approach are effective for the architecture design (see Fundamental theories lesson in section 7.1.3). The first manner is ensuring that the architecture includes processes that force a CDSS to collaborate with clinical end users (QAR1, QAR2). It depends on the contextual characteristics to what extent partnership is desired. Design principle 1 (subsection 8.1.1) allows for the customization of the level of updating automation. However, an architecture should always avoid the implementation of a CDSS that fully excludes clinical end users. Moreover, the architecture should guard against unsolicited influence from the CDSS provider (QAR2). This kind of influence would have a similar effect on the perceived control of clinical end users regarding a CDSS.

The second manner is to ensure the architecture gives end users control over the change in the recommendation generator model. Therefore, the architecture should describe a model update

engine that is sensitive to what clinical end users find important (DGR7). Design principle 1 covers this sensitivity to end users' preferences (subsection 8.1.1). However, by letting end users determine which choices an update incorporates, the architecture does not guarantee the update provides a recommendation generator model that the end user desires. Therefore, the clinical end user should have the power to reject a model update and reset the active model by replacing it with an older model update (QAR4). Table 8.6 gives an overview of the requirements fundamental to design principle 5.

Table 8.6: *Architecture Requirements design principle 5: Receptivity to user input.*

Reference	Architecture Requirement
QAR1	The architecture should always contain processes that work in partnership with clinical end users.
QAR4	The architecture should avoid the development of a CDSS that poses a model version to the clinical end user.
QAR2	The architecture should minimize the number of intervening actions needed from Council that are not requested by a clinical end user.
DGR7	The architecture should force the development of a CDSS that estimates new parameters according to the choice types and weight specification clinical end users selected as soon as the clinical end user deems this model inaccurate or undesired for decision support.

8.1.6 Design principle 6: Differentiation in consumed choices and produced information

Design principle: The architecture should distinguish types of choices that different CDSS processes consume and types of information that a CDSS produces for different receivers.

The architecture should differentiate types of information a dynamic BAIT-based CDSS consumes and produces. Without this requisite, an architecture designer might overlook the differences between types of information. The result may be that the architecture describes a CDSS that stores information in the most efficient way possible. For instance, by storing all choices as being of one type. However, a successful architecture design requires a more sophisticated storage of information. The need for this differentiation in information types stems from two reasons. First of all, a dynamic BAIT-based CDSS architecture describes processes that consume different types of choices. The list below gives three examples of processes that require a differentiation in choice types:

1. The model update engine includes two processes that both use different types of choices (DGR1,DGR8). The estimation process consumes both experiment choices and real-life choices. The validation process only consumes real-life choices (see Problem Framing lesson 3 in section 7.1.1).
2. Within the validation process, the construction of the training set and validation set requires the separation of choices based on the patient number. For some patients, multiple physicians make a choice (see section 6.3.1). By doing so, the choice base in the architecture stores multiple judgements for similar choice tasks. The architecture should describe a model update engine that ensures that choices concerning the same patient are equally divided over the training and validation set (see Fundamental theories lesson 3 in section 7.1.3. When the training set or the validation set contains very similar choices, the validation process will generate a biased performance assessment.
3. When a CDSS consists of an Information specification Extension, a subset of choices influences an update of the recommendation generator model differently than the choices outside that subset (see subsection 4.3.1). For instance, if end users want an update to be more influenced by senior end users' choices than juniors end users' choices. Therefore, the architecture should separate choices in terms of their features (DGR3). Examples of features are the choice source (experiment choice or real-life choice), the date of choice, the recommendation of a CDSS, the confidence of the decision-maker, the experience level of the clinical end user).
4. While a subset of choices influences the estimation of the recommendation generator model during an update (see the above-mentioned bullet point), the validation set needs to consist

of choices from the original choice base that stores the complete set of choices (DGR8). If the validation set uses the choices in the temporary choice base, the set will contain duplicates. When the validation set contains duplicates, the validation process will generate a biased performance assessment.

Second, the components in the architecture produce information that is of interest to different receivers. The architecture should ensure a CDSS presents the produced information either to the CDSS provider, end users, or to both (DGR11, DGR12). Moreover, a CDSS should distinguish information for different receivers to protect end users' privacy (see Emerging artefact lesson 4 in section 7.1.2 and subsection 8.1.9). To ensure a CDSS presents the correct information to the correct receiver, the architecture should differentiate the produced information according to the intended receiver. As a result, the architecture describes multiple information flows with different sources and different receivers. To avoid the development of CDSSs with unintended information flows, the architecture should emphasize both which components consume information and produce information and the associated dependencies between these components (CAR5, CAR9). Table 8.7 gives an overview of the requirements fundamental to design principle 6.

Table 8.7: *Architecture Requirements design principle 6: Differentiation in consumed choices and produced information.*

Reference	Architecture Requirement
DGR1	The architecture should force the development of a CDSS that distinguishes experiment choices and real-life choices.
DGR8	The architecture should force the development of a CDSS that assesses the performance of a newly estimated choice recommendation generator model based on a unique set of recent real-life choices.
DGR3	The architecture should force the development of a CDSS that stores a choice with all features assigned to the choice when a clinical end user entered the choice into the CDSS.
CAR5	The architecture should distinguish components that produce information and components that consume information.
CAR9	The architecture should inform on all dependencies between architecture components.
DGR11	The architecture should force the development of a CDSS that enables clinical end users to request the performance metrics for all choice recommendation generator model updates.
DGR12	The architecture should force the development of a CDSS that gives Council insight into the performance metrics for the choice recommendation generator model updates of all healthcare contexts.

8.1.7 Design principle 7: Mutual learning

Design principle: The architecture should force the development of a CDSS that enhances mutual learning between clinical end users.

As subsection 8.1.5 defines, clinical end users value their autonomy. A consequence of this autonomy is that clinical end users tend to be confident about their judgement and choice resulting from this judgement. On the contrary, clinical end users are fundamentally interested in learning from the decision-making of other clinical end users. However, clinical end users lack a tool that triggers them to set aside their autonomy and engage in mutual learning. A BAIT-based CDSS closes this gap because a BAIT-based CDSS can make the decision-making behaviour of a pool of clinical end users visible (see Problem framing lesson 1 in section 7.1.1). A dynamic BAIT-based CDSS can even do this over time, making the evolution of clinical end users' decision-making behaviour explicit. As a result, a dynamic BAIT-based CDSS has the characteristics that allow clinical end users to investigate each other's decision-making behaviour and, by doing so, learn from each other over time.

To exploit this learning potential, the architecture should describe components that allow clinical end users to request and investigate the information reflecting the decision-making behaviour in a decision-making context (DGR15). Moreover, the architecture should include components that will enable clinical end users to request an overview of a particular subgroup's decision-making behaviour and the evolution of this group's behaviour (DGR16). By doing so, a senior clinical end user can inform his or her decision-making strategy with the decision-making behaviour of junior clinical end users. A clinical end user from a specific discipline can learn about the decision-making strategy of a clinical end user from another discipline. However, this functionality is bound to privacy considerations. Section 8.1.9 covers these privacy considerations. Table 8.8 gives an overview of the requirements fundamental to this design principle 7.

Table 8.8: *Architecture Requirements design principle 7: Mutual learning.*

Reference	Architecture Requirement
DGR15	The architecture should force the development of a CDSS that allows clinical end users to request the relative importance of the choice attributes of all model updates.
DGR16	The architecture should force the development of a CDSS that allows clinical end users to request the relative importance of the choice attributes for a by the clinical end user selected subgroups.

8.1.8 Design principle 8: Architecture intuitiveness

Design principle: The architecture should have an intuitive design for developers with knowledge and skill in Discrete Choice Modeling.

Intuitive design means that when a developer sees the architecture, the developer understands what to do. An intuitive architecture design is realized in two ways. The first way is to ensure the architecture eases the load of research and design validation activities. The architecture should present all the building blocks as straight-forward as possible, so no additional efforts are required from the developer to prepare the CDSS development (CAR7, CAR13).

Besides easing the burden of additional efforts, the architecture should give a comprehensible description in a uniform way. The architecture should realize this by addressing three challenges related to the characteristics of a dynamic BAIT-based CDSS architecture. The first characteristic is that a dynamic BAIT-based CDSS contains many components and information flows. Therefore, the architecture of a dynamic BAIT-based CDSS will be complex. When the architecture gives an ambiguous description of the CDSS components, developers will develop unintended CDSS's. As a result, the architecture may lead to CDSS's that are unreliable. To avoid that developers can interpret the architecture in multiple ways, the architecture designer should take care of three aspects.

1. The designer should ensure that the architecture explicates everything that is not directly clear in a static representation (CAR2). An architecture is only a static representation that does not show all functionalities that need to occur during the run time of the CDSS. For instance, some processes need to run in parallel because their outcomes are mutually dependent.
2. The architecture designer should use a single description language (CAR4).
3. The designer should complement the architecture with two additional tools. The first tool is a set of guidelines. The guidelines explain the description language and the reasoning behind design choices fundamental to the architecture (CAR 6). By doing so, the guidelines enhance a CDSS developer's understanding of the architecture. The second tool is a data object overview. To reduce the complexity of the architecture, the architecture should not describe every single data object. The definition of many different data objects - for example, single values - makes the architecture information-dense. Because the data objects are vital for the intended CDSS development, all data objects and the databases that store these objects need to be defined in a data object overview that complements the architecture (CAR8).

The second characteristic is that a dynamic BAIT-based CDSS architecture is adaptable (see subsection 8.1.1). Therefore, the architecture consists of core and optional components. The architecture must clarify which elements are fundamental and which are negotiable. The third characteristic is that the architecture includes human-dependent and human-independent processes. To avoid a CDSS developer overlooks essential interaction elements, the architecture should emphasize the components involve interaction with the end user (CAR3).

A notion regarding design principle 8 is necessary. The components that the architecture describes heavily rely on Discrete Choice Modeling (DCM) theories and practices. Therefore, the architecture designer should assume that the developer who will use the architecture has basic knowledge of DCM and that the design is intuitive for developers familiar with the basic concepts of DCM. Table 8.9 gives an overview of the requirements fundamental to design principle 8.

Table 8.9: *Architecture Requirements design principle 8: Architecture intuitiveness.*

Reference	Architecture Requirement
CAR13	The architecture should define a CDSS that is implementable within four weeks.
CAR4	The architecture should be designed according to one description language.
CAR2	The architecture should mark parallel processes that have to run at the same time.
CAR3	The architecture should mark processes that require collaboration between a clinical end user and a CDSS.
CAR6	The architecture should inform how the architecture is used for development in specific healthcare decision-making contexts.
CAR7	The architecture should inform on all processes that are needed to achieve the goal of the CDSS.
CAR8	The architecture should inform on all data objects associated with the architecture.

8.1.9 Design principle 9: Privacy of choice information and decision-making behaviour information

Design principle: The architecture should not contain decision-making information flows that can be traced back to individual clinical end users or are undesired by clinical end users in the healthcare context.

The application of a dynamic BAIT-based CDSS evokes additional and different privacy issues than traditional CDSSs (see Fundamental theories lesson 1 in section 7.1.3). A dynamic BAIT-based CDSS continuously consumes choice information and produces decision-making behaviour information. As a consequence, the architecture deals with three types of privacy considerations, each involving a different set of stakeholders (see Emerging artefact lesson 4 in section 7.1.2). To protect the privacy of the information at all three levels, the architecture designer should respect two types of requisites. The first requisite is that the architecture should not include information flows that receivers can assign to individual clinical end users. The first requisite is that the architecture must ensure the CDSS developer can skip the implementation of information flows that end users might define as undesired in the Information Processing Agreement (see Emerging artefact lesson 3 in section 7.1.2).

The requirement covering the privacy of the information captured by a dynamic BAIT-based CDSS has a relation with other architecture requirements (see section 8.2). However, this requirement is not combined with other requirements to avoid that architecture designers will overlook the requirement. As an illustration, design principle 6 covers the separation of decision-making information to be accessed by different receivers. Therefore, it is related to the requirement that concerns information privacy. However, the fulfilment of design principle 6 does not guarantee the protection of the privacy of that information. Therefore, an additional design principle needs to be in place that encourages architecture designers to ascertain the protection of end user's privacy. Table 8.10 presents the requirement fundamental to design principle 9.

Table 8.10: *Architecture Requirements Design principle 9: Privacy of choice information and decision-making behaviour information.*

Reference	Architecture Requirement
CAR14	The architecture should ensure choices are not retrievable to an individual clinical end user and Council only has access to decision-making behaviour for which end users provided permission.

8.1.10 Design principle 10: Explication of the organizational activities for the CDSS provider

Design principle: The architecture should be complemented with a description of the organizational activities needed from the CDSS provider and a procedure that guides the provider in executing these activities.

An implemented dynamic BAIT-based CDSS requires the ongoing involvement of the CDSS provider. Consequently, the success of a CDSS is closely related to the service that the CDSS provider provides (see Emerging artefact lesson 5 in section 7.1.2). This service includes a series of

organizational activities. Because the architecture should inform a developer on all processes that need to be in place to achieve the goals of a CDSS (CAR7), the architecture should also inform the developer about the organizational activities a CDSS requires.

The architecture may not be clear to a CDSS provider who needs to execute the activities. Therefore, the architecture should be complemented with a description of all organizational tasks for CDSS providers. This description should assist providers in executing the required organizational activities, anticipating these activities, and fitting the activities into the existing workflow of the CDSS provider (CAR6, CAR7, CAR12). As an illustration, the organizational activity description should define the steps a CDSS provider should perform to ensure the CDSS developer is informed about the context-specific preferences of clinical end users or to provide end users with support on the use of a CDSS. Table 8.11 gives an overview of the requirements fundamental to this design principle.

Table 8.11: *Architecture Requirements design principle 10: Explication of the organizational activities.*

Reference	Architecture Requirement
CAR6	The architecture should inform how the architecture is used for development in specific healthcare decision-making contexts.
CAR7	The architecture should inform on all processes that are needed to achieve the goal of the CDSS.
CAR12	The architecture should always be integrable with the service environment of Council.

8.2 The relations within the set of design principles

The design principles that section 8.1 describes are interdependent in multiple ways. Five interrelations are prominent.

1. Design principles 4 and 5 (see Figure 8.1). Both design principles reflect the importance of involving the end user. A dynamic BAIT-based CDSS can involve end users in two ways. The first is by making the internal change of a CDSS's recommendation generator model comprehensible and tractable for end users. The second is by making a CDSS receptive to user input so end users can control the internal change of a CDSS's recommendation generator model. An architecture designer should design for user involvement while considering both focus areas. To encourage a designer to make complementary design decisions concerning both focus areas, the research distinguishes design principles 4 and 5.

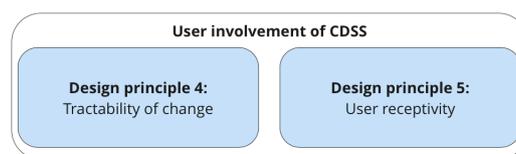


Figure 8.1: Design principle 4 and design principle 5 both enhance a CDSS's user involvement.

2. Design principles 1 and 5 (see Figure 8.2). The realization of design principle 5 requires adherence to design principle 1. Design principle 5 forces the receptivity of clinical end users by allowing end users to control the internal change of a CDSS's recommendation generator model. The architecture allows end users to shape this internal change with Information specification Extensions (see subsection 4.3.1). The design of these extensions stems from design principle 1 (see subsection 8.1.1).
3. Design principles 3 and 5 (see Figure 8.2). Design principle 3 poses a restriction on the realization of design principle 5. Although clinical end users should be able to interact with a CDSS, this interaction should always align with the clinical end users' goals. Therefore, the architecture designer should be careful with describing information flows from a CDSS to clinical end users.
4. Design principles 1, 7 and 9 require adherence to design principle 6 (see Figure 8.3). Principle 6 ensures that the architecture distinguishes between types of information. Without making

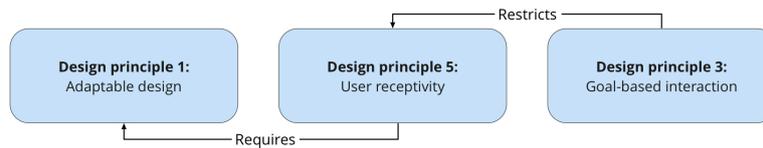


Figure 8.2: Design principle 5 relates to design principle 3 and design principle 1.

this distinction, a designer will not be able to successfully implement the components required to adhere to design principles 1, 7, and 9. For instance, design principle 1 and 7 require the architecture to let end users specify subgroups of choices. Therefore, an architecture should distinguish choices in terms of their features. Finally, the protection of the privacy of choice information and decision-making behaviour information (design principle 9) requires the architecture to distinguish information that a CDSS can share with particular receivers from information a CDSS cannot share with particular receivers.

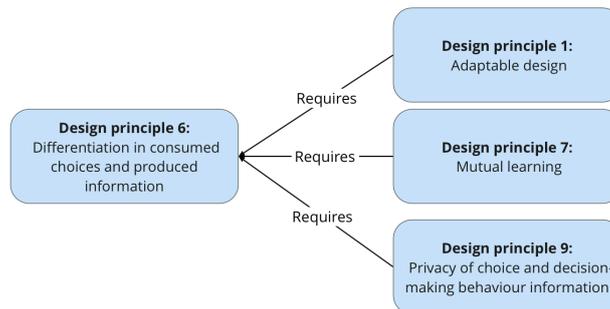


Figure 8.3: Design principle 6 supports design principles 1, 7 and 9.

- Design principles 7 and 9 (see Figure 8.4). Principle 7 enhances mutual learning by generating insight into the decision-making behaviour of specific subgroups of end users. Principle 9 puts a restriction on mutual learning as soon as the categorization into subgroups leads to decision-making behaviour information that is retrievable to an individual end user.

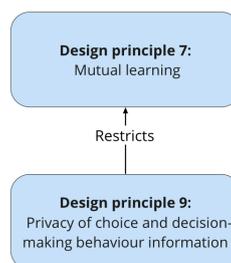


Figure 8.4: Design principle 9 restricts design principle 7.

The relationships between the design principles indicate that the ten design principles for the design of a dynamic BAIT-based CDSS together form a coherent set of design principles.

8.3 Contribution to architecture design science knowledge

The ten design principles form a coherent set of guidelines for future designers. This section compares the design principles for designing a dynamic BAIT-based CDSS architecture with the guidelines that already exist in architecture design literature. By doing so, this section makes explicit the similarities and differences between designing an architecture of a dynamic BAIT-based CDSS and a traditional dynamic CDSS architecture. As a result, the section explicates the novelty of the design principles and proves the contribution of the design principles to the architecture design and CDSS architecture design knowledge base.

8.3.1 Contribution design principle 1: Adaptable design

The use of an adaptable design that contains various extensions is not often mentioned in the context of CDSS architecture design (Xiao, Cousins, Fahey, Dimitrov, & Hederman, 2012). (Edwin, 2014) states that designers avoid this adaptable structure because it increases the complexity of an architecture. The studies that do mention the use of an adaptable structure focus on the modifiability of the architecture to reflect contextual changes in the CDSS implementation (Xiao et al., 2012). However, no suggestions on or examples of CDSS architectures were found that include extensions to meet varying client needs.

In the context of a dynamic BAIT-based CDSS, the extensions enable customization of the choices informing updates. CDSS architecture design literature does not describe architectures with extensions or CDSS updating components that enable clinical end users to customize the information that shapes updates. Traditional CDSSs commonly are black-boxes (non-knowledge-based CDSSs) or have stifle techniques (knowledge-based CDSSs). These CDSSs do not allow for customized weighting of information in an update. Therefore, architecture designers cannot design extensions for the updating. Moreover, designs often focus on finding the optimal level of updating automation rather than letting it be context-dependent (El-Sappagh & El-Masri, 2011, 2014; Greenes et al., 2018; Xiao et al., 2012). As a result, there is an absence of principles guiding towards an adaptable CDSS architecture that allows for a customized updating process. Therefore, design principle 1 forms a novel contribution to the CDSS architecture design knowledge base.

The use of a layered architecture design approach is already widely suggested in architecture design literature and forms an established practice (Clements, Garlan, Little, Nord, & Stafford, 2003; Shaw & Garlan, 1996). It even is the earliest architectural style that has been ever used (Savolainen & Myllarniemi, 2009). Most architecture description languages, like ArchiMate, propose a layered design approach. Beyond some exceptions with more detailed layers, existing literature commonly proposes three layers: the business layer, application layer, and technology layer (for an explanation, see subsection 4.1.1). Some design studies or methods refer to the layers with a different name. Also in the context of CDSSs, this layered architecture approach is often used (Cho, Kim, Kim, Kim, & Kim, 2010; Kim, Cho, & Kim, 2008; Kumar, 2015; Oh et al., 2015; Y.-F. Zhang et al., 2016).

However, because the usage of a layered approach for architecture design significantly varies in different contexts (Savolainen & Myllarniemi, 2009), it is of value to investigate whether it also is beneficial in the context of a BAIT-based CDSS architecture. Design principle 2 provides a confirmation. However, adopting a layered approach in the context of a dynamic BAIT-based CDSS differs in two ways from existing architecture designs.

1. Although a layered approach is beneficial for an architecture's flexibility, flexibility is not often mentioned as a reason for the choice of a layered approach (Savolainen & Myllarniemi, 2009). Because of the set of extensions the architecture should include, architectural flexibility will be the foremost reason for a layered approach in the context of a dynamic BAIT-based CDSS.
2. Most of the architecture designs suggest a technology layer and emphasize its importance (Fan et al., 2017; Hussain, Afzal, Khan, & Lee, 2012). The reason is that traditional CDSSs need to extract data from various data sources and need a connection to the hospital information systems (HIS) or Electronic Health Record (EHR) (Esmailzadeh et al., 2015; Goldberg et al., 2016; Oh et al., 2015). This technology layer especially needs further specification in a dynamic context, where new data must be continuously retrieved from, produced for, and shared with other sources. The operation of a dynamic BAIT-based CDSS does not rely on other systems in the healthcare context. Therefore, the architecture can describe all components and information flows representing the production and consumption of information a dynamic BAIT-based CDSS uses with software processes and data objects at the application layer. As a result, the structure of a dynamic BAIT-based CDSS should also be defined with a layered architecture, but there are no considerations to present at the technology layer.

To finish, one may argue that an adaptable structure is unnecessary in the context of a layered approach because a layered approach is often used as a means for modifiable architectures by defining "loosely coupled" components. However, the layered approach does not emphasize the definition of prefabricated building blocks from which end users can choose to customize the information shaping updates. Therefore, the suggestion of a layered approach would fall short in the context of a dynamic BAIT-based CDSS architecture since a layered design does not propose a set of optional extensions.

8.3.2 Contribution design principle 2: Objectivity maximization within the subjective boundaries

CDSS architecture literature does not explicitly mention the balance between objectivity and subjectivity of an update. This absence is probably related to the novelty of the extensions that introduce subjectivity. Therefore, the balance between an objective and subjective update must be particularly taken into account when designing a dynamic BAIT-based CDSS architecture. In addition, the realization of an objective update was found to slightly different in the context of a dynamic BAIT-based CDSS. Although reliability is an established value in architecture and CDSS design literature (Vogel, Arnold, Chughtai, & Kehrer, 2011), its realization in the context of a dynamic BAIT-based CDSS requires the designer to deliberately reuse Discrete Choice Modeling (DCM) and Machine Learning (ML) techniques. Architecture design literature does not mention the deliberate reuse of these techniques. Moreover, a dynamic BAIT-based CDSS architecture designer should beware of the personal influence. A dynamic BAIT-based CDSS processes single real-life choices entered by individuals instead of large bulks of historical data like ML models. By doing so, the structure of a BAIT-based CDSS allows for design alternatives that nurture personal influence. Therefore, it is important that a designer is aware of these alternatives and carefully takes distance from them. However, design recommendations concerning the mitigation of personal influence were not found in CDSS architecture design work.

8.3.3 Contribution design principle 3: Goal-based interaction

As illustrated by (Castillo & Kelemen, 2013; Gretton, 2018; Pirnejad et al., 2019; Varonen et al., 2008), the interaction between a CDSS and a clinical end user should be effective and brief for the CDSS not to interrupt or even frustrate the end user. This research shows that a dynamic BAIT-based CDSS architecture designer should also focus on balancing and prioritizing the type and timing of information that drives the interaction. A balance was found in ensuring that all information provided and asked has a clear purpose from the point of view of the clinical end user. Besides this confirmation of what architecture design literature already describes,

A dynamic BAIT-based CDSS generates information and interaction on an ongoing basis. As a result, the architecture of a dynamic BAIT-based CDSS deals with more information compared to traditional CDSSs. Consequently, a dynamic BAIT-based CDSS architecture designer needs to put extra effort into ensuring the interaction between a CDSS and a clinical end user will not interrupt or frustrate clinical end users.

8.3.4 Contribution Design principle 4: Tractability of change

CDSS architecture and CDSS design research widely acknowledge the importance of transparency (Gretton, 2018; Melton et al., 2016; Siau & Shen, 2003; Rawson et al., 2017; Røst et al., 2020; Wang et al., 2021). Literature frames transparency as the quality that allows a CDSS to explain its choice recommendation to the end user. Despite that a dynamic BAIT-based CDSS directly reflects the changes in the clinical end user's own expert knowledge, clinical end users still want a dynamic BAIT-based CDSS to make transparent how it processes these changes for an update. Design principle 4 contributes by emphasizing that architecture designers should ensure that the architecture of a dynamic BAIT-based CDSS describes a CDSS that makes the changes in the recommendation generator model transparent to end users.

Although the importance of transparency is acknowledged, no CDSS designs can provide enough explanatory information on a recommendation and on how severe each recommendation is (Wang et al., 2021). CDSS design studies rather think of the wish for transparency as a threat for the adoption and users' acceptance than as a design opportunity (Khairat et al., 2018). A plausible reason is that the technologies in which the state-of-the-art CDSS's are grounded restrict the design space for transparent CDSSs over time. Non-knowledge-based (NKB) CDSSs are opaque by nature (Wang et al., 2021) (for an explanation, see section 1.1). These CDSSs cannot make the processes and rules followed to generate a recommendation explicit (Khairat et al., 2018; Shaikh et al., 2020). As a result, the design space for transparency is restricted. This restriction makes the definition of design principles that guide the design of a transparent operation meaningless. Knowledge-based (KB) CDSSs can make the decision-making rules they follow to generate a choice recommendation transparent but are inflexible (for an explanation, see section 1.1). Because of this inflexible character, KB CDSSs cannot make the subtle changes in the operation transparent over time. KB CDSSs can only approach it by presenting the differences between scripts with if-then-else rules over time. The closest attempt to make the operation transparent was found in

the research of Khairat et al. (2018) Khairat et al. (2018) propose an approach with which the CDSS presents the rules that the CDSS followed to generate a recommendation. However, this requires the clinical end user to understand the full process and to spend a large amount of time to compare all rules used by the recommendation generation processes.

On the contrary, a dynamic BAIT-based CDSS does possess the characteristics to satisfy the wish for a dynamic CDSS to make the changes within in a recommendation generator model transparent. Therefore, defining a design principle that guides the design of this transparency is worthwhile for the first time. Moreover, having the opportunity to design for this transparency also raised the opportunity to investigate what requisites a dynamic CDSS that makes its operation transparent over time must meet. Finally, the research findings show that designing a transparent dynamic CDSS concerns the complete structure of a CDSS, including all components and information flows.

To conclude, the contribution of the findings associated with design principle 4 consists of three points:

- The design principle shows that a transparent updating operation is also greatly valued in the context of a CDSS that directly translates the expertise and knowledge of its clinical end users.
- The design principle shows that it is worthwhile to define a design principle on the transparent design in the context of a BAIT-based CDSS.
- The design principle gives insight into the considerations relevant when designing an architecture for a dynamic CDSS that makes the changes in the recommendation generator model transparent.
- The design principles guides designers to deem transparency as a value that concerns the complete structure of a CDSS rather than a individual component.

8.3.5 Contribution design principle 5: Receptivity to user input

Prior work has already proved that the perceived threat of a CDSS to an end user's professional autonomy and to an end user's ability to control the CDSS directly affects their willingness to use a CDSS (Esmailzadeh et al., 2015; Friedberg et al., 2014; Sambasivan et al., 2012; Wang et al., 2021). Moreover, design principles that encourage designers to design for user control have been in place for a long time (Nielsen & Molich, 1990). However, CDSS design literature frames this threat as if it stems from CDSS components that provide recommendations and make choices for the clinical end user (Esmailzadeh et al., 2015). Therefore, literature online provides evidence and examples that make designers aware of the threat when designing CDSS tasks that directly need the knowledge and skill of clinical end users.

The findings of this research show that clinical end users also prefer to control activities that do not directly replace their knowledge and skill. Design principle 5 contributes by emphasizing that a designer of a dynamic BAIT-based CDSS architecture should ensure that a dynamic CDSS is receptive to control commands from clinical end users, although the CDSS mainly involves model management activities that do not require expert knowledge or skill.

8.3.6 Contribution design principle 6: Differentiation in choices

Because of the novelty of the combination of DCM and CDSS architecture design, no design principles or other guiding evidence on dealing with choices as source of information are found in CDSS architecture design literature yet. As such, the importance of keeping choices of a particular source, collected at a specific point in time and with particular features separate was not emphasized before. This was thereby found to be important in the particular context of a BAIT-based CDSS architecture.

8.3.7 Contribution design principle 7: Mutual learning

CDSS architecture design literature does not discuss mutual learning as a functionality of a CDSS. The reason is that the BAIT approach offers a dynamic BAIT-based CDSS with three unique characteristics that enable mutual learning in a dynamic setting:

1. The CDSS codifies the knowledge of the pool of clinical end users in the particular context

2. The CDSS can store the features of choices (for instance, the experience of the choice-maker in terms of being a senior or a junior clinical end users). By doing so, the CDSS can expose the decision-making behaviour of subgroups.
3. The CDSS can visualize how the decision-making behaviour (of subgroups) develops over time.

When designing an architecture for a dynamic BAIT-based CDSS, the designer should exploit these characteristics. By doing so, the designer ensures the architecture describes information flows and includes processes that enable clinical end users to learn from each other over time.

Mutual learning was a greatly valued phenomenon that provides insights and constitutes learning activities that cannot be realized otherwise. Although its power and importance, mutual learning is not the head goal of a CDSS. Consequently, a designer may easily overlook it. Therefore, design principle 7 forms an indispensable contribution to CDSS architecture design knowledge.

8.3.8 Contribution design principle 8: Architecture intuitiveness

Architecture design literature does not mention the concept of architecture intuitiveness as such. However, architecture design literature does emphasize concepts with a similar focus, like architecture understandability, consistency, and clarity (Alenezi, 2016; Ali, Baker, O’Crowley, Herold, & Buckley, 2018; Perry & Wolf, 1992; Shahin, Liang, & Khayyambashi, 2010; Yu, Breslau, & Shenker, 1999). Along with the research, it was found that these concepts are also essential qualities for an architecture of a dynamic BAIT-based CDSS.

However, the reasons why these qualities are important for a dynamic BAIT-based CDSS architecture slightly differ from the reasons for generic architectures. First of all, most architecture studies present the architecture as a tool to communicate and illustrate the features of the CDSS rather than as an artefact that is to be reused by different developers within a CDSS provider organization. Because the architecture of a dynamic BAIT-based CDSS should be useful as a guiding tool for various developers, the architecture’s understandability, consistency, and clarity are even more important. Second, because the design of a dynamic BAIT-based CDSS architecture involves combining discrete choice modeling (DCM), Machine Learning (ML) and CDSS practices and theories, it contains references to different knowledge fields and is highly complex. Therefore, the architecture must be completely understandable, consistent, and clear. Finally, it contains optional extensions. The architecture should make clear that these extensions are not mandatory.

By just being understandable, the architecture of a dynamic BAIT-based CDSS does not tackle all of the challenges mentioned above. Therefore, the concept of intuitiveness was chosen to summarize the features that an architecture designer must realize when designing a dynamic BAIT-based CDSS architecture. Although the term intuitive mainly summarizes what literature already expresses, the design principle does contribute to architecture design knowledge. The principle does so by combining all relevant architecture features for an architecture that needs to function as a finalized artefact instead of as a means in a software development process. In case of the latter, the architecture designer is not explicitly concerned with features of the architecture. Instead, the architecture designer uses the architecture to find the correct CDSS structure and focuses on the requirements of the CDSS. For an explanation, see section 7.1.4. Moreover, it contributes by emphasizing that intuitiveness is to be judged by a developer with knowledge and skill in DCM. Therefore, this design principle is strongly related to the particular case of a dynamic BAIT-based CDSS architecture that deals with DCM theories and practices.

8.3.9 Contribution design principle 9: Privacy of choices and decision-making behaviour information

Because many architecture design and CDSS design studies design for privacy, privacy is an established value to take into account when designing an architecture (Hoepman, 2014; Rubinstein & Good, 2013; Wilk et al., 2013; Vogel et al., 2011). Therefore, it was not expected that a design principle would be dedicated to privacy. However, it was found that the design of a dynamic BAIT-based CDSS architecture does benefit from a design principle guiding them.

An architecture of a dynamic BAIT-based CDSS deals with information representing end user’s choices and decision-making behaviour. The choice information and the decision-making behaviour information were found to be privacy-sensitive. Traditional CDSS architectures do not deal with these types of information. Consequently, existing evidence and findings may provide insufficient guidance to guarantee the protection of these particular types of information. Therefore, design

principle 9 contributes by guiding the design for privacy protection in the specific context of a dynamic BAIT-based CDSS architecture.

8.3.10 Contribution design principle 10: Explication of the organizational activities

Complementing an architecture with guiding documents is established practice (Leist & Zellner, 2006; Vogel et al., 2011). Both architecture design methods and best practices found in architecture design literature argue that the architecture should be delivered with a package of reference materials (Rouhani, Mahrin, Nikpay, Ahmad, & Nikfard, 2015; Rozanski & Woods, 2012). A lack of supporting material on the use of the architecture and its implementation might result in CDSS's that are ineffective or unintended (Rouhani et al., 2015). For the architecture design of a dynamic BAIT-based CDSS this complementary guiding material was also found to be indispensable.

However, this principle makes an additional note that was not explicitly found in CDSS architecture design. Because the architecture defines a dynamic CDSS that requires ongoing service from the CDSS provider, the materials should define the organizational activities that a CDSS provider should execute. These activities are indispensable to guarantee a successful application of a developed dynamic BAIT-based CDSS.

8.4 Summary chapter 8

The goal of chapter 8 is to make the move towards the generalization of the research findings. With this purpose, the tested architecture requirements were translated into design principles that form recommendations on how future designers should design dynamic BAIT-based CDSS architectures.

The design principles result from clustering the architecture requirements that effectively outline a dynamic BAIT-based CDSS architecture for application in a situated context. These requirements are strongly interrelated. According to the relations between the requirements, the requirements were grouped into overarching themes that represent design principles. The lessons learned identified during an ongoing reflection on the design process informed this clustering process. As a result, the design principles form representations of that what is true - the architecture requirements - and that what is important to do to avoid the obstacles encountered during this research process. By doing so, the learning from the situated design project is developed into a general solutions concept for a class of field problems. The result is the following set of ten design principles:

1. **Adaptable design:** The architecture should have a technology-independent, adaptable structure with extensions that enable customization of the updating automation level and the choice information that updates incorporate.
2. **Objectivity maximization within the subjective boundaries:** The architecture should maximize the objectivity with which choice information is processed during an update, given the clinical end user's deliberately chosen updating preferences.
3. **Goal-based interaction:** The architecture should only define interaction flows that are always available and from which a clinical end user directly or indirectly benefits.
4. **Tractability of change:** The architecture should only describe CDSS components and outcomes that are comprehensible for clinical end users and that make the change in the choice recommendation generator model tractable.
5. **Receptivity to user input:** The architecture should force a receptive CDSS that induces interaction with end users and give end users control over the change in the choice recommendation generator model.
6. **Differentiation in consumed choices and produced information:** The architecture should distinguish types of choices that different CDSS processes consume and types of information that a CDSS produces for different receivers.
7. **Mutual learning:** The architecture should force the development of a CDSS that enhances mutual learning between clinical end users.
8. **Architecture intuitiveness:** The architecture should have an intuitive design for developers with knowledge and skill in Discrete Choice Modeling.

9. Privacy of choice information and decision-making behaviour information: The architecture should not contain decision-making information flows that can be traced back to individual clinical end users or are undesired by clinical end users in the healthcare context.
10. Explication of the organizational activities: The architecture should be complemented with a description of the organizational activities needed from the CDSS provider and a procedure that guides the provider in executing these activities.

The principles are interdependent. The interdependency denotes that the design principles form a coherent set of principles that together guide the design of future dynamic BAIT-based CDSS architectures. Some design principles confirm the importance of existing design principles in the particular context of a dynamic BAIT-based CDSS architecture. Other design principles capture design guidelines that are specifically important for the design of a dynamic BAIT-based CDSS architecture. These principles are novel and form a contribution to the CDSS architecture design knowledge base.

Chapter 9

Conclusion

The main goal of this research is to develop design principles for the design of an architecture that functions as a tool for the development of dynamic BAIT-based Clinical Decision Support Systems (CDSS) for various and ever-changing healthcare decision-making contexts. This chapter presents the main findings (section 9.1), the scientific contributions to theory (section 9.2), the identified limitations, and recommendations on future research (section 9.3). The outcomes of this research answer the following question:

To what design principles should a system architecture of a dynamic BAIT-based CDSS adhere?

9.1 Main findings: design principles for a dynamic BAIT-based CDSS architecture

The answer to the main research question consists of ten design principles that guide the design of an architecture of a dynamic BAIT-based CDSS. The design principles build onto the main components that CDSS architecture design literature suggests for developing CDSSs in dynamic healthcare decision-making contexts: an adaptive knowledge base so that a CDSS accepts new information, a model update engine so a CDSS's recommendation generator model continually bases its recommendations on that new information, a model update monitor so that the CDSS timely detects any decrease in performance in a changing context, and a Human-Computer Interaction (HCI) component. The design of these components, however, closely relates to the technological basis of existing CDSSs.

However, this technological basis fundamentally differs from the BAIT approach, which applies Discrete Choice Modeling (DCM) to codify expert knowledge of clinical end users operating the CDSS. Therefore, DCM theories, practices, and characteristics delimit the design of the main dynamic CDSS components. Consequently, an architecture designer cannot directly reuse the existing components in the context of a dynamic BAIT-based CDSS. To find out what the design of these components looks like in the context of a dynamic BAIT-based CDSS, this research followed the Action Design Research (ADR) framework. The ADR framework makes it possible to identify the requirements of a dynamic BAIT-based CDSS architecture and to test these requirements by building an architecture in a situated problem context. The final set of requirements outline an architecture that CDSS developers in the situated context can use to build dynamic BAIT-based CDSSs for various healthcare decision-making contexts. The formalization of the situated architecture requirements resulted into ten design principles that guide future designers who aim to develop a dynamic CDSS-based architecture.

The set of design principles include seven main insights regarding the design of a dynamic BAIT-based CDSS architecture. The first insight is that the architecture of a dynamic BAIT-based CDSS should enable customization of the level of updating automation and the types of choices that inform a CDSS's model update engine. By doing so, the architecture enables CDSS developers to satisfy end users' preferences regarding a dynamic BAIT-based CDSS in various healthcare contexts. Beyond this subjectivity, the architecture designer should ensure that the architecture maximizes the objectivity with which the update engine reflects the changes in decision-making contexts. To this end, the designer should exclude design alternatives that evoke personal influence on a CDSS's model update engine and only apply modified Machine Learning (ML) techniques that

work reliably in a dynamic BAIT-based CDSS. The third insight is that a designer should ensure that the architecture only defines information flows that ask information from clinical end users or provide clinical end users with information from which they directly or indirectly benefit. Fourth, the architecture should ensure an implemented dynamic BAIT-based CDSS involves clinical end users in two ways:

1. The architecture should only incorporate CDSS components and outcomes that are comprehensible for clinical end users and that make the change in a CDSS's recommendation generator model tractable for clinical end users. Although a dynamic BAIT-based CDSS directly reflects the changes in the clinical end user's expert knowledge, end users are not likely to accept a dynamic CDSS if it does not make transparent how it processes these changes during an update.
2. The architecture should ensure a dynamic BAIT-based CDSS induces interaction with clinical end users and allows end users to control the change in a CDSS's recommendation generator model. The latter ensures that a CDSS bases its recommendations on information the end users find relevant.

Fifth, a designer should ensure that the architecture enhances mutual learning between a pool of clinical end users. Clinical end users greatly value mutual learning. The BAIT approach gives a dynamic CDSS the unique capability to make explicit how clinical end users' decision-making behaviour evolves. Insight into this evolving behaviour constitutes mutual learning activities. Because this insight stems from the BAIT approach, end users cannot engage in these activities without a dynamic BAIT-based CDSS. Therefore, the architecture should utilize this unique capability and describe a CDSS that enhances mutual learning between clinical end users.

The sixth insight is that the architecture should distinguish the types of choice information a dynamic BAIT-based CDSS consumes, and the types of information that the CDSS produces. The separation of choice types is necessary because each CDSS process consumes different types of choices. The separation in the information that the CDSS produces is necessary to manage the accessibility of information by different end users and the CDSS provider. As a result, the architecture guarantees the protection of privacy within healthcare contexts.

The final finding is that the success of a dynamic BAIT-based CDSS relies not only on technical components but also on the service that a CDSS provider offers. In the context of a dynamic BAIT-based CDSS, this service provision includes a series of organizational activities that the provider should execute on an ongoing basis. To guide the CDSS provider in performing these activities, the architecture should be complemented with guidelines informing the CDSS provider on the to be expected organizational efforts.

9.2 Theoretical contributions of the design research

The leading contribution provides new knowledge to the CDSS architecture design knowledge base. This contribution is of interest to architecture designers who want to develop transparent and dynamic CDSS architectures based on Discrete Choice Modeling (DCM) theories and practices. The contribution is threefold: a set of novel design principles, a foundation for dynamic BAIT-based CDSS architecture design research, and reusable solutions concepts. Moreover, a theoretical contribution is dedicated to the Machine Learning (ML) knowledge base from which theories and practices were ingrained into the architecture design.

9.2.1 Design principles

The ten design principles form an important theoretical contribution to CDSS architecture design knowledge for two reasons. First of all, the lessons learned during the ongoing reflection on the design process inspired the formulation of the design principles. Therefore, the principles capture what designers should do to avoid the obstacles encountered during this research process. Second, the principles guide CDSS architecture designers. Some principles confirm the usefulness of principles CDSS architecture already suggest in the particular context of a dynamic BAIT-based CDSS architecture. More important, the research identified novel design principles that are specific for designing a dynamic BAIT-based CDSS architecture. These principles provide new knowledge. The list below summarizes the most prevalent novel contributions to CDSS architecture design knowledge:

- There was an absence of design principles promoting and guiding an adaptable CDSS architecture design with extensions that allow clinical end users to customize the updating process. Moreover, it is common practice to design a CDSS architecture with a business, an application, and a technology layer. However, the principles inform a designer that the architecture can leave any software and hardware requisites at the technology layer unspecified. By doing so, the principles provide novel information that is specific for the design of a dynamic BAIT-based CDSS architecture.
- For the first time, the design principles require the designer to keep an eye on the balance between an objective and subjective reflection of contextual changes in an update. Besides, the principles guide a designer in realizing this objectivity with design considerations that are not straightforward and not explicitly mentioned in CDSS architecture design literature.
- CDSS architecture and CDSS design research widely acknowledge the importance of transparency. However, there are no technologies that enable the design of transparent and dynamic CDSSs. As a result, evidence and examples of designing for transparency in the context of dynamic CDSS are absent. The introduction of a dynamic BAIT-based CDSS makes it worthwhile to define a design principle for CDSS architecture designers that concerns transparency. Moreover, there was space to explore and test alternatives that facilitate transparency in dynamic CDSSs for the first time. As a result, the design principles contribute by explicating how designers can develop an architecture that describes a dynamic CDSS that is transparent.
- Prior work already argues that the professional autonomy of end users is an important consideration in CDSS design. However, this work focuses on the threat posed by CDSS tasks that directly require the knowledge and skill of clinical end users. The set of design principles contributes by defining that the designer should also ensure that the CDSS is receptive to input from clinical end users for tasks that do not directly challenge the knowledge and skill of clinical end users but rather involve the tasks of a dynamic CDSS, like updating and quality monitoring.
- No design principles concerning information in the form of choices are in place yet. Therefore, architecture design literature did not emphasize the importance of distinguishing types of choices before.
- Because mutual learning is unique for a dynamic BAIT-based CDSS, existing CDSS architecture design literature does not guide designing for mutual learning. Therefore, the principles form an indispensable contribution to CDSS architecture design knowledge.
- Privacy is an established value in architecture design literature. However, because a dynamic BAIT-based CDSS consumes choice information and produces decision-making behaviour information, existing evidence and examples may provide insufficient guidance. As a result, the principle guiding the design for privacy in the context of a BAIT-based CDSS architecture forms a contribution to the privacy guidelines that are already in place.

9.2.2 Foundation for BAIT-based CDSS architecture design work

The second contribution of the research findings to CDSS architecture design is the foundation for the design of transparent and dynamic CDSS architectures. Thus far, the development of these architectures was restrained by the lack of guidance in the challenges associated with designing a dynamic BAIT-based CDSS architecture. The design is challenging because the BAIT approach is a novel technological innovation. Therefore, the technical and social feasibility in dynamic healthcare decision-making contexts is still to be proved. Moreover, designing the architecture requires combining theories and practices from Discrete Choice Modeling (DCM), CDSS architecture design, and Machine Learning (ML). These knowledge areas were not combined before. The design also requires starting without any building blocks in place. Finally, the architecture design needs to incorporate the preferences of clinical end users in varying healthcare contexts that may also change over time.

The lessons learned during this research and the resulting design principles support CDSS architecture designers in tackling these challenges. By doing so, the lessons and design principles fill the lack of guidance. It was shown how the BAIT approach can be successfully used in dynamic contexts and how the theories and practices of DCM, CDSS and ML can be intertwined. Moreover, there are fundamental architecture building blocks in place, and there are manners found by which

the varying preferences of healthcare contexts can be satisfied. Accordingly, for the first time CDSS architecture designers are in the position to further develop CDSS architectures that enable the development of transparent and dynamic CDSSs that generate accurate recommendations to clinical end users in ever changing decision-making contexts.

9.2.3 Reusable solution concepts

The collection of reusable solution concepts forms the third contribution to CDSS architecture design. The architecture has an adaptable design with optional extensions. The way the architecture is designed enables designers to reuse individual components and or the architecture as a whole.

Reusing the complete architecture

The architecture was found effective for the development of CDSSs that will be applied in dynamic healthcare contexts. Contexts where the knowledge fundamental to a decision-making strategy changes, such as governmental institutions or the pharmaceutical industry, are dynamic. Therefore, CDSS providers or developers benefit from the architecture when they aim to develop CDSSs for application in these contexts. However, the architecture can also be used for CDSSs that are to be applied in static contexts. Instead of updating a dynamic CDSS according to contextual change, the updates of a dynamic CDSS aim at a better understanding of the static context.

Something specific to healthcare contexts is the presence of professional autonomy. To allow end users to control the CDSS, the architecture maximizes the value of transparency. The professional autonomy and the perceived control over the CDSS may be less important in other dynamic contexts. For example, in governmental contexts where many simple decisions are made on a repetitive basis. However, the end users in these contexts may still value the focus on transparency in the architecture. As illustration, a governmental institution needs to explain the reasoning behind the decisions that affect civilians. Therefore, the institution needs to be able to track the changes in the updated support model they consult to make the decisions. The transparent approach of the architecture is still of value: not directly to the end user, but indirectly to the group of individuals served by the end user.

Reusing the individual components

The architecture provides the main dynamic CDSS components that are transformed to match the technological foundation of a BAIT-based CDSS. The main components are the adaptive choice base, model update engine, model quality monitor, recommendation generator, model management module, and user settings. The model update engine is especially useful in other contexts where choices involve human characteristics because it deals with the challenges of estimating and validating a DCM choice model with patient data, like the inability of replacing missing values cannot be replaced. However, the update engine will be less useful when the contextual decision-making conditions change quickly and the decision-making deals with deep uncertainty. In deep uncertain contexts, the update engine will never change the recommendation generator model with new knowledge quickly enough, so it perfectly understands significantly different choice tasks. In addition, reusing the update engine is meaningless for the design of CDSSs for application in contexts where only one decision-maker is active. The update engine will estimate recommendation generator models that entirely rely on choices made by this individual.

Finally, the architecture defines reusable components that are optional. These optional extensions allow for the customization of the CDSS. The architecture consists of two types of extensions. First, the architecture describes three extensions that allow for the modification of the information that is included in an update. Therefore, they are all applicable in a wide variety of contexts: in contexts where experiment choices are considered important because of the noisy real-life setting or where choices made in real-life should be assigned with more weight because they rarely occur. Second, the architecture includes three extensions that allow customizing the level of updating automation. These variants can be effectively reused in contexts where end users would first like to develop trust in the CDSS, before they let the CDSS do the work. For instance, in contexts where decisions impact human well-being or where professional autonomy plays an important role. An example beyond the healthcare sector is the juridical sector.

9.2.4 Contribution to Machine Learning knowledge field

Machine Learning (ML) theories and practices are ingrained in the design of the architecture. ML thereby provided important building blocks during the design process. However, while applying

these theories and practices, it was found that ML theories are not directly applicable in the context of dynamic CDSSs. The reuse of these theories is complicated for the following two main reasons. Many ML techniques do not distinguish samples in a data set. However, the updates of a dynamic CDSS incorporate information about real-life patients. This patient information may include extreme cases that are perceived as inappropriate representations to inform an update of a CDSS. Therefore, techniques that consume this information need to distinguish unequal samples. Second, to assess the performance of a dynamic CDSS it should be determined how well the CDSS can recommend end users in the present context. Validation techniques suggested by ML mainly focus on assessing the performance of a model for all data samples used for the model construction. By doing so, these techniques do not consider the recentness of the samples. This research suggests the manipulation of the ML k-fold validation to assess the performance of a CDSS in the present situation.

Moreover, it was found that the black-box characteristic of ML is a real issue for application in healthcare contexts. Because of the opaque nature of ML models, it seems designers commonly believe that designing for transparency is pointless. Consequently, barely any attempts on making ML-based CDSSs more transparent were encountered. Encouraging designers to approach transparency would, however, pay off. Along with this research, it was found that clinical end users already more value a CDSS that makes parts of the model training and testing transparent. For instance, by giving insight into the samples that the CDSS used for the model validation. This insight allows end users to understand and judge the value of the performance assessment better.

9.3 Limitations and Recommendations on further research

This sections presents the limitations of the research outcomes and does suggestions for further research.

9.3.1 Limitations of the design principles

The first limitation is that the design principles do not guide the design of a dynamic CDSS that reflects radical changes in the context. An example of a radical change is the introduction of a new treatment that influences end users' decision-making. Second, the design principles stem from the design process conducted by a single designer. Because of the designer's basic knowledge in DCM and architecture design, the formulation of the design principles assumes that the user of the principles has basic knowledge about the key theories and practices of both fields. Moreover, the principles guide the expansion of the current BAIT-based CDSS. Therefore, they assume that a designer understands the architecture of the current BAIT-based CDSS. Third, the application of a dynamic BAIT-based CDSS is bound to contexts with a limited level of contextual change. In contexts where conditions change very quickly, but there is no clue how they will change, a dynamic BAIT-based CDSS will not be able to be prepared with the accurate knowledge in time. The design principles are thereby only applicable in contexts where the factors influencing decision-making are foreseeable.

Finally, due to resource restrictions and physicians' busy schedules, only three clinical end users were involved in the requirement identification. Therefore, the interview results may have provided a biased picture of both the important requirements. In addition, the architecture defines an expansion of a CDSS that is already in place. As a result, interviewed end users who use the current BAIT-based CDSS may have overlooked CDSS aspects that the static BAIT-based CDSS already captures but are still relevant for the dynamic CDSS. For instance, the clinical end users did not go into the reliability of the components of the dynamic BAIT-based CDSS. To avoid that the interview findings resulted in a incomplete set of requirements, the research included several measures concerning the requirement identification. The first measure is that the involved clinical end users are all different in type. By doing so, the interviews give insight into different viewpoints on the dynamic BAIT-based CDSS. Second, representatives of Council who know about the preferences of a broader pool of clinical end users were interviewed. Third, the requirement identification was complemented with a literature review. The literature review covered the important features that were not mentioned during the interviews. However, because of the novelty of the BAIT-based CDSS and its application in dynamic contexts, it cannot be proved that the completion with literature is satisfying.

9.3.2 Recommendations for further research

There are three categories of recommendations for further research: research efforts to improve the architecture design (1), research to answer new design questions that stem from the findings of this research (2), and research to address knowledge gaps encountered during this research (3). For each category, this section briefly explains the recommendations. For three recommendations, this section gives a more extensive research plan.

1. Recommended design efforts to improve the architecture

The architecture design is associated with three improvement points, which give rise to three suggestions for further research. First of all, the architecture assumes a continuous inflow of real-life choices, which is not feasible in each decision-making context. Further research should investigate alternative manners to collect real-life choices to realize a sufficient inflow of real-life choices in any context. Second, the architecture does not allow for the incorporation of radical change as subsection 9.3.1 explains. Ensuring a dynamic BAIT-based CDSS incorporates radical change is challenging because it requires adjustments in the recommendation generator model's structure. An adjustment may be adding an attribute to the model, removing an attribute from the model, or modifying the value range of an attribute. Accounting for these adjustments in the CDSSs of all end users requires too much effort from a CDSS provider. However, there is no alternative in place that allows clinical end users without DCM knowledge to make these adjustments themselves. Moreover, a procedure needs to be in place that explains how to translate different types of radical change into a new model structure. Examples of types of radical change are a new insight into a treatment's side-effects and an adjustment in medical rules fundamental to the decision-making. Further research is needed to design solutions that address the challenges associated with the incorporation of radical change.

A final improvement of the architecture requires research on the inclusion of objective clinical outcomes (OCO's) in the updating process. By doing so, the CDSS improves its recommendations based on the results of choices made. The inclusion of OCO's turned out to be of great value to clinical end users. However, the incorporation of OCO's in the architecture is complicated. First of all, because there is a lack of objective evaluation: clinical end users may differ in their judgement on what outcome forms a good or bad result. Second, because the inflow of OCO's is unpredictable. In some healthcare contexts, the treatment and the recovery take a long time. To tackle these challenges, research investigating what criteria a CDSS should meet to include OCO's in the updating process is needed. This research could involve a series of case studies in different healthcare decision-making contexts to identify what requirements a CDSS should satisfy to incorporate OCO's. For instance, to identify what types of mechanisms are needed to evaluate outcomes and to identify how OCO's should be weighted compared to other decision information during an update. When the criteria are at hand, subsequent research needs to focus on designing the architecture components according to these criteria.

2. New design challenges

The findings of this research provided insight into three new design challenges. The recommendations on researching these challenges aim at encouraging research on BAIT-based CDSS architecture design so that the quality of transparent and dynamic CDSSs will improve over time. First of all, because clinical end users move during their working days, they will benefit from a mobile application. Further design research could therefore use the design principles to design a similar architecture for a mobile-based CDSS that an end user can consult from everywhere. Second, designers are encouraged to take the design principles and investigate how well they work for designing an architecture in contexts other than the healthcare sector. The results should argue if and how the design principles must be adjusted, replaced, or removed for application in other contexts.

Third, a design challenge involves designing an architecture with a model update engine that can deal with interdependent choices. For instance, if the choice for a particular treatment depends on the previous choices made in a patient's medical history. Designing a dynamic BAIT-based CDSS architecture that covers the interdependency between choices is challenging for two reasons. First, a designer needs to clarify the different dependencies between choices. These choice dependencies may vary over healthcare contexts or even over sectors. Second, the designer needs to determine how the estimation and validation processes should be modified to account for all the dependencies in various contexts. To do so, a designer will need to consult econometric and statistical theories.

The list below suggests three research steps for a designer to incorporate the interdependency of choices in a dynamic BAIT-based CDSS architecture:

1. Defining how treatment choices are related to each other in a set of different contexts.
2. Formulating criteria that a BAIT-based CDSS must meet to incorporate the choice dependencies.
3. Design one or multiple update engine(s) that meet the formulated criteria.

3. Research to fill identified knowledge gaps

Finally, this research indicates three fundamental knowledge gaps worth researching. Filling in the first gap requires the design of a framework. DCM proposes various decision-making rules that describe how individuals make decisions, like the Random Utility Maximization and Random-Regret rule. A BAIT-based CDSS can generate recommendations following all different rules. Because the combination of DCM and CDSS design is new, it is unclear how these rules function for decision support in varying contexts. A framework that provides insight into the connections between particular decision-making rules that a BAIT-based CDSS can follow to generate choice recommendations and the characteristics of different application contexts could fill this gap. The design of this framework will require two research steps. First, the researcher should define a short-list of decision-making rules. Subsequently, the researcher should determine for which contextual characteristics each rule forms a suitable representation of the decision-making strategy.

Second, CDSS design literature shows that a higher level of transparency of the decision support enhances the clinical end users' feeling of control. The more control clinical end users have over a CDSS, the less likely it is that the end users perceive a CDSS as a threat to their professional autonomy. However, it is not quantified to what extent the transparent character of a dynamic BAIT-based CDSS can mitigate the threat to clinical end users' autonomy, whether this is enough to increase the likelihood of acceptance, and what other criteria a CDSS must meet to be perceived less as a threat by clinical end users.

The final gap is methodological. ADR is a relatively new research method, and no CDSS architecture design studies following the ADR framework exist yet. This research shows the benefits of ADR for CDSS architecture design, like the continuous reflection on and improvement of the architecture. Other architecture design studies also recognize the benefits of ADR. However, the design of a system architecture differs from that of other IT artefacts. The architecture design requires considerations at two levels: the system and architecture level. To encourage architecture designers to exploit the benefits of ADR, the barrier to use ADR should be lower. For instance, by modifying the building, implementation, and evaluation stage. This modification could involve adding steps that guide designers in dealing with the requirement identification and evaluation at the two levels. This modification could also complement the set ADR principles with architecture-specific design principles. Further research needs to investigate how to customize the ADR framework so that it tackles the challenges central to architecture design research.

References

- Abbasi, M., & Kashiyarndi, S. (2010). *Clinical decision support systems: A discussion on different methodologies used in health care. report*, 1–15.
- Abowd, G., Allen, R., & Garlan, D. (1993). Using style to understand descriptions of software architecture. *ACM SIGSOFT Software Engineering Notes*, 18(5), 9–20.
- Abu-Mostafa, Y., Magdon-Ismael, M., & Lin, H. (2012). *Learning from data vol. 4: Amlbook new york. NY, USA*.
- Afzal, M., Hussain, M., Khan, W. A., Ali, T., Lee, S., & Kang, B. H. (2014). Knowledgebutton: An evidence adaptive tool for cdss and clinical research. In *2014 ieee international symposium on innovations in intelligent systems and applications (inista) proceedings* (pp. 273–280).
- Agarwal, R., & Tanniru, M. R. (1990). Knowledge acquisition using structured interviewing: an empirical investigation. *Journal of Management Information Systems*, 7(1), 123–140.
- Akour, M., Aldiabat, S., Alsgaier, H., Alkhateeb, K., & Alenezi, M. (2016). Software architecture understandability of open source applications. *International Journal of Computer Science and Information Security*, 14(10), 65.
- Alenezi, M. (2016). Software architecture quality measurement stability and understandability. *International Journal of Advanced Computer Science and Applications (IJACSA)*, 7(7), 550–559.
- Alexander, G. L. (2006). Issues of trust and ethics in computerized clinical decision support systems. *Nursing administration quarterly*, 30(1), 21–29.
- Ali, N., Baker, S., O’Crowley, R., Herold, S., & Buckley, J. (2018). Architecture consistency: State of the practice, challenges and requirements. *Empirical Software Engineering*, 23(1), 224–258.
- Allen, R., & Garlan, D. (1994). Formalizing architectural connection. In *Proceedings of 16th international conference on software engineering* (pp. 71–80).
- Alshammry, T. F., & Alenezi, M. (2017). Software architecture understandability in object-oriented systems. *i-Manager’s Journal on Software Engineering*, 12(2), 1.
- AlSharif, M., Bond, W. P., & Al-Otaiby, T. (2004). Assessing the complexity of software architecture. In *Proceedings of the 42nd annual southeast regional conference* (pp. 98–103).
- Alwosheel, A. S. A. (2020). Trustworthy and explainable artificial neural networks for choice behaviour analysis.
- Anderson, J. A., & Willson, P. (2008). Clinical decision support systems in nursing: synthesis of the science for evidence-based practice. *CIN: Computers, Informatics, Nursing*, 26(3), 151–158.
- Angehrn, A. A., & Lüthi, H.-J. (1990). Intelligent decision support systems: a visual interactive approach. *Interfaces*, 20(6), 17–28.
- Anooj, P. (2012). Clinical decision support system: Risk level prediction of heart disease using weighted fuzzy rules. *Journal of King Saud University-Computer and Information Sciences*, 24(1), 27–40.
- Aoki, N. (2020). An experimental study of public trust in ai chatbots in the public sector. *Government Information Quarterly*, 37(4), 101490.
- Applegate, L., McFarlan, F., & McKenney, J. (1996). *Corporate information systems management: Text and cases (chapter 12)*. Chicago, IL: Irwin.
- Arnott, D. (2006). Cognitive biases and decision support systems development: a design science approach. *Information Systems Journal*, 16(1), 55–78.
- Aron, R., Dutta, S., Janakiraman, R., & Pathak, P. A. (2011). The impact of automation of systems on medical errors: evidence from field research. *Information systems research*, 22(3), 429–446.
- Arsanjani, A. (2004). Service-oriented modeling and architecture. *IBM developer works*, 1, 15.

- Asokan, G., & Asokan, V. (2015). Leveraging “big data” to enhance the effectiveness of “one health” in an era of health informatics. *Journal of epidemiology and global health*, 5(4), 311–314.
- Axsen, J., Mountain, D. C., & Jaccard, M. (2009). Combining stated and revealed choice research to simulate the neighbor effect: The case of hybrid-electric vehicles. *Resource and Energy Economics*, 31(3), 221–238.
- Baba, N., & Suto, H. (2000). Utilization of artificial neural networks and the td-learning method for constructing intelligent decision support systems. *European Journal of Operational Research*, 122(2), 501–508.
- Babar, M. A., Winkler, D., & Biffi, S. (2007). Evaluating the usefulness and ease of use of a groupware tool for the software architecture evaluation process. In *First international symposium on empirical software engineering and measurement (esem 2007)* (pp. 430–439).
- Balzer, W. K., Doherty, M. E., et al. (1989). Effects of cognitive feedback on performance. *Psychological bulletin*, 106(3), 410.
- Bayramzadeh, S., Joseph, A., Allison, D., Shultz, J., Abernathy, J., & Group, R. O. S. (2018). Using an integrative mock-up simulation approach for evidence-based evaluation of operating room design prototypes. *Applied ergonomics*, 70, 288–299.
- Bech, M. (2003). Politicians’ and hospital managers’ trade-offs in the choice of reimbursement scheme: a discrete choice experiment. *Health policy*, 66(3), 261–275.
- Beck, M. J., Fifer, S., & Rose, J. M. (2016). Can you ever be certain? reducing hypothetical bias in stated choice experiments via respondent reported choice certainty. *Transportation Research Part B: Methodological*, 89, 149–167.
- Behmel, S., Damour, M., Ludwig, R., & Rodriguez, M. (2021). Intelligent decision-support system to plan, manage and optimize water quality monitoring programs: design of a conceptual framework. *Journal of Environmental Planning and Management*, 64(4), 703–733.
- Bell, P., Hoadley, C. M., & Linn, M. C. (2004). Design-based research in education. *Internet environments for science education, 2004*, 73–85.
- Ben-Akiva, M., Bradley, M., Morikawa, T., Benjamin, J., Novak, T., Oppewal, H., & Rao, V. (1994). Combining revealed and stated preferences data. *Marketing Letters*, 5(4), 335–349.
- Ben-Akiva, M., McFadden, D., Abe, M., Böckenholt, U., Bolduc, D., Gopinath, D., ... others (1997). Modeling methods for discrete choice analysis. *Marketing Letters*, 8(3), 273–286.
- Ben-Akiva, M., & Morikawa, T. (1990). Estimation of switching models from revealed preferences and stated intentions. *Transportation Research Part A: General*, 24(6), 485–495.
- Ben-Akiva, M. E., McFadden, D., Train, K., et al. (2019). *Foundations of stated preference elicitation: Consumer behavior and choice-based conjoint analysis*. Now.
- Bengio, Y., & Grandvalet, Y. (2004). No unbiased estimator of the variance of k-fold cross-validation. *Journal of machine learning research*, 5(Sep), 1089–1105.
- Bennett, C. C., & Doub, T. W. (2016). Expert systems in mental health care: Ai applications in decision-making and consultation. In *Artificial intelligence in behavioral and mental health care* (pp. 27–51). Elsevier.
- Bharosa, N., & Janssen, M. (2015). Principle-based design: a methodology and principles for capitalizing design experiences for information quality assurance. *Journal of Homeland Security and Emergency Management*, 12(3), 469–496.
- Bhat, C. R., & Castelar, S. (2002). A unified mixed logit framework for modeling revealed and stated preferences: formulation and application to congestion pricing analysis in the san francisco bay area. *Transportation Research Part B: Methodological*, 36(7), 593–616.
- Bhatt, G. D., & Zaveri, J. (2002). The enabling role of decision support systems in organizational learning. *Decision Support Systems*, 32(3), 297–309.
- Bierlaire, M., & Fetiaron, M. (2009). Estimation of discrete choice models: extending biogeme. In *Swiss transport research conference (strc)*.
- Billings, C. E. (2018). *Aviation automation: The search for a human-centered approach*. CRC Press.
- Birol, E., Kontoleon, A., & Smale, M. (2006). *Combining revealed and stated preference methods to assess the private value of agrobiodiversity in hungarian home gardens*. Intl Food Policy Res Inst.
- Björnander, S. (2011). Architecture description languages. *Mrtc. Mdh. Se*.
- Blank, T., Graves, K., Sepucha, K., & Llewellyn-Thomas, H. (2006). Understanding treatment decision making: contexts, commonalities, complexities, and challenges. *Annals of Behavioral Medicine*, 32(3), 211–217.
- Boehm, B., Bose, P., Horowitz, E., & Lee, M. J. (1995). Software requirements negotiation and renegotiation aids: A theory-w based spiral approach. In *1995 17th international conference*

- on software engineering (pp. 243–243).
- Boxall, P. C., Adamowicz, W. L., Swait, J., Williams, M., & Louviere, J. (1996). A comparison of stated preference methods for environmental valuation. *Ecological economics*, 18(3), 243–253.
- Boyce, D. E., & Williams, H. C. (2015). *Forecasting urban travel: Past, present and future*. Edward Elgar Publishing.
- Brehmer, B. (1992). Dynamic decision making: Human control of complex systems. *Acta psychologica*, 81(3), 211–241.
- Brownstone, D., Bunch, D. S., & Train, K. (2000). Joint mixed logit models of stated and revealed preferences for alternative-fuel vehicles. *Transportation Research Part B: Methodological*, 34(5), 315–338.
- Brownstone, D., & Small, K. A. (2005). Valuing time and reliability: assessing the evidence from road pricing demonstrations. *Transportation Research Part A: Policy and Practice*, 39(4), 279–293.
- Buchgeher, G., & Weinreich, R. (2014). Continuous software architecture analysis. In *Agile software architecture* (pp. 161–188). Elsevier.
- Burrell, J. (2016). How the machine ‘thinks’: Understanding opacity in machine learning algorithms. *Big Data & Society*, 3(1), 2053951715622512.
- Byrnes, C., & Kyrtatzoglou, I. (2006). *Applying architecture tradeoff assessment method (atam) as part of formal software architecture review* (Tech. Rep.). MITRE CORP BEDFORD MA BEDFORD United States.
- Campbell, D., Hutchinson, W. G., & Scarpa, R. (2006). Lexicographic preferences in discrete choice experiments: Consequences on individual-specific willingness to pay estimates.
- Carlsson, F., Kataria, M., & Lampi, E. (2010). Dealing with ignored attributes in choice experiments on valuation of sweden’s environmental quality objectives. *Environmental and resource economics*, 47(1), 65–89.
- Carson, R. T., & Groves, T. (2007). Incentive and informational properties of preference questions. *Environmental and resource economics*, 37(1), 181–210.
- Castaneda, C., Nalley, K., Mannion, C., Bhattacharyya, P., Blake, P., Pecora, A., . . . Suh, K. S. (2015). Clinical decision support systems for improving diagnostic accuracy and achieving precision medicine. *Journal of clinical bioinformatics*, 5(1), 1–16.
- Castillo, R. S., & Kelemen, A. (2013). Considerations for a successful clinical decision support system. *CIN: Computers, Informatics, Nursing*, 31(7), 319–326.
- Checkland, P., & Scholes, J. (1990). *Soft systems methodology in action*. (No. Q295 C51).
- Chen, E. S., Borlowsky, T., Qureshi, K., Li, J., Lussier, Y. A., & Hripcsak, G. (2007). Monitoring the function and use of a clinical decision support system. In *Amia annu symp proc* (Vol. 902).
- Cherchi, E., & Hensher, D. A. (2015). Workshop synthesis: Stated preference surveys and experimental design, an audit of the journey so far and future research perspectives. *Transportation Research Procedia*, 11, 154–164.
- Chikwe, J. E. (n.d.). Decision-making feasibility and techniques: A psychological and strategic evaluation imperatives. *GLOBAL JOURNAL OF BUSINESS MANAGEMENT*, 1.
- Cho, I., Kim, J., Kim, J. H., Kim, H. Y., & Kim, Y. (2010). Design and implementation of a standards-based interoperable clinical decision support architecture in the context of the korean ehr. *International journal of medical informatics*, 79(9), 611–622.
- Chorus, C. G. (2010). A new model of random regret minimization. *European Journal of Transport and Infrastructure Research*, 10(2).
- Clements, P., Bass, L., Kazman, R., & Abowd, G. (1995). Predicting software quality by architecture-level evaluation. In *Proceedings of the fifth international conference on software quality* (Vol. 5, pp. 485–497).
- Clements, P., Garlan, D., Little, R., Nord, R., & Stafford, J. (2003). Documenting software architectures: views and beyond. In *25th international conference on software engineering, 2003. proceedings*. (pp. 740–741).
- Cole, R., Puro, S., Rossi, M., & Sein, M. (2005). Being proactive: where action research meets design research. *ICIS 2005 proceedings*, 27.
- Cummings, M. L. (2006). Automation and accountability in decision support system interface design.
- Curcin, V., Fairweather, E., Danger, R., & Corrigan, D. (2017). Templates as a method for implementing data provenance in decision support systems. *Journal of biomedical informatics*, 65, 1–21.
- Danaf, M., Becker, F., Song, X., Atasoy, B., & Ben-Akiva, M. (2019). Online discrete choice models: Applications in personalized recommendations. *Decision Support Systems*, 119,

- Dasgupta, S., et al. (1996). *Technology and creativity*. Oxford University Press, USA.
- de Freitas, L. M., Becker, H., Zimmermann, M., & Axhausen, K. W. (2019). Modelling intermodal travel in Switzerland: A recursive logit approach. *Transportation Research Part A: Policy and Practice*, 119, 200–213.
- Degtjar, I., & Rose, S. (2021). A review of generalizability and transportability. *arXiv preprint arXiv:2102.11904*.
- Delen, D. (2019). *Prescriptive analytics: The final frontier for evidence-based management and optimal decision making*. FT Press.
- de Luca, S., & Cantarella, G. E. (2009). *Validation and comparison of choice models*. Ashgate, UK.
- DeShazo, J., & Fermo, G. (2004). *Implications of rationally-adaptive pre-choice behaviour for the design and estimation of choice models*. University of California, Los Angeles.
- Ding, M., Grewal, R., & Liechty, J. (2005). Incentive-aligned conjoint analysis. *Journal of marketing research*, 42(1), 67–82.
- Di Noia, T., Mongiello, M., Nocera, F., & Straccia, U. (2019). A fuzzy ontology-based approach for tool-supported decision making in architectural design. *Knowledge and Information Systems*, 58(1), 83–112.
- Dissanayake, D., & Morikawa, T. (2003). A combined rp/sp nested logit model of vehicle ownership, mode choice and trip chaining to investigate household travel behavior in developing countries. In *Trb 2003 annual meeting cd-rom, nagoya*.
- Djamasbi, S., & Loiacono, E. T. (2008). Do men and women use feedback provided by their decision support systems (dss) differently? *Decision Support Systems*, 44(4), 854–869.
- Dym, C., Little, P., Orwin, E., & Spjut, R. (2004). *Engineering design: A project based approach*. Hoboken, NJ: Wiley.
- Eapen, B. (2021). *Towards a theory of adoption and design for clinical decision support systems* (Unpublished doctoral dissertation).
- Edwards, W. (1962). Dynamic decision theory and probabilistic information processings. *Human factors*, 4(2), 59–74.
- Edwin, N. M. (2014). Software frameworks, architectural and design patterns. *Journal of Software Engineering and Applications*, 2014.
- El-Sappagh, S. H., & El-Masri, S. (2011). A proposal of clinical decision support system architecture for distributed electronic health records. In *Proceedings of the international conference on bioinformatics & computational biology (biocomp)* (p. 1).
- El-Sappagh, S. H., & El-Masri, S. (2014). A distributed clinical decision support system architecture. *Journal of King Saud University-Computer and Information Sciences*, 26(1), 69–78.
- Erder, M., & Pureur, P. (2015). *Continuous architecture: sustainable architecture in an agile and cloud-centric world*. Morgan Kaufmann.
- Esmailzadeh, P., Sambasivan, M., Kumar, N., & Nezakati, H. (2015). Adoption of clinical decision support systems in a developing country: Antecedents and outcomes of physician’s threat to perceived professional autonomy. *International journal of medical informatics*, 84(8), 548–560.
- Fan, A., Lin, D., & Tang, Y. (2017). Clinical decision support systems for comorbidity: Architecture, algorithms, and applications. *International journal of telemedicine and applications*, 2017.
- Feit, E. M., Beltramo, M. A., & Feinberg, F. M. (2010). Reality check: Combining choice experiments with market data to estimate the importance of product attributes. *Management science*, 56(5), 785–800.
- Folmer, E., Van Gurp, J., & Bosch, J. (2004). Software architecture analysis of usability. In *Ifip international conference on engineering for human-computer interaction* (pp. 38–58).
- Fong, J. (2001). *Knowledge management & intelligent enterprises*. World Scientific.
- Franses, P. H. (2000). A test for hit rate in binary response models. *International Journal of Market Research*, 42(2), 1–5.
- Friedberg, M. W., Chen, P. G., Van Busum, K. R., Aunon, F., Pham, C., Caloyeras, J., . . . others (2014). Factors affecting physician professional satisfaction and their implications for patient care, health systems, and health policy. *Rand health quarterly*, 3(4).
- Fryszak, J. (2016). A preliminary framework for feedback mechanism design in extended decision support systems-implications of a literature review. In *Mcis* (p. 37).
- Fu, K. K., Yang, M. C., & Wood, K. L. (2015). Design principles: The foundation of design. In *International design engineering technical conferences and computers and information in engineering conference* (Vol. 57175, p. V007T06A034).

- Gago, P., & Santos, M. F. (2008). Towards an intelligent decision support system for intensive care units. In *Workshop on supervised and unsupervised ensemble methods and their applications* (p. 21).
- Garg, A. X., Adhikari, N. K., McDonald, H., Rosas-Arellano, M. P., Devereaux, P. J., Beyene, J., ... Haynes, R. B. (2005). Effects of computerized clinical decision support systems on practitioner performance and patient outcomes: a systematic review. *Jama*, *293*(10), 1223–1238.
- Garlan, D. (2000). Software architecture: a roadmap. In *Proceedings of the conference on the future of software engineering* (pp. 91–101).
- Garlan, D., & Shaw, M. (1993). An introduction to software architecture. In *Advances in software engineering and knowledge engineering* (pp. 1–39). World Scientific.
- Gaube, S., Suresh, H., Raue, M., Merritt, A., Berkowitz, S. J., Lerner, E., ... Ghassemi, M. (2021). Do as ai say: susceptibility in deployment of clinical decision-aids. *NPJ digital medicine*, *4*(1), 1–8.
- Gefen, D., Karahanna, E., & Straub, D. W. (2003). Trust and tam in online shopping: An integrated model. *MIS quarterly*, 51–90.
- Gluud, C., & Nikolova, D. (2007). Likely country of origin in publications on randomised controlled trials and controlled clinical trials during the last 60 years. *Trials*, *8*(1), 1–8.
- Goldberg, H. S., Paterno, M. D., Grundmeier, R. W., Rocha, B. H., Hoffman, J. M., Tham, E., ... others (2016). Use of a remote clinical decision support service for a multicenter trial to implement prediction rules for children with minor blunt head trauma. *International journal of medical informatics*, *87*, 101–110.
- Gong, Y. (2012). Engineering flexible and agile services: a reference architecture for administrative processes.
- Gorton, I. (2011). Software quality attributes. In *Essential software architecture* (pp. 23–38). Springer.
- Goździł-Mitka, I., & Okreglicka, M. (2014). Improving decision making in complexity environment. *Procedia Economics and Finance*, *16*, 402–409.
- Greefhorst, D., & Proper, E. (2011). The role of enterprise architecture. In *Architecture principles* (pp. 7–29). Springer.
- Greene, W. (2009). Discrete choice modeling. In *Palgrave handbook of econometrics* (pp. 473–556). Springer.
- Greenes, R. A., Bates, D. W., Kawamoto, K., Middleton, B., Osheroff, J., & Shahar, Y. (2018). Clinical decision support models and frameworks: seeking to address research issues underlying implementation successes and failures. *Journal of biomedical informatics*, *78*, 134–143.
- Gretton, C. (2018). Trust and transparency in machine learning-based clinical decision support. In *Human and machine learning* (pp. 279–292). Springer.
- Grimm, K. J., Mazza, G. L., & Davoudzadeh, P. (2017). Model selection in finite mixture models: A k-fold cross-validation approach. *Structural Equation Modeling: A Multidisciplinary Journal*, *24*(2), 246–256.
- Grossglauer, M., & Saner, H. (2014). Data-driven healthcare: from patterns to actions. *European journal of preventive cardiology*, *21*(2_suppl), 14–17.
- Gultepe, E., Green, J. P., Nguyen, H., Adams, J., Albertson, T., & Tagkopoulos, I. (2014). From vital signs to clinical outcomes for patients with sepsis: a machine learning basis for a clinical decision support system. *Journal of the American Medical Informatics Association*, *21*(2), 315–325.
- Hacker, P., Krestel, R., Grundmann, S., & Naumann, F. (2020). Explainable ai under contract and tort law: legal incentives and technical challenges. *Artificial Intelligence and Law*, 1–25.
- Haider, W. (2002). Stated preference and choice models—a versatile alternative to traditional recreation research. In *Monitoring and management of visitor flows in recreational and protected areas. conference proceedings* (pp. 115–121).
- Haj-Bolouri, A., Puro, S., Rossi, M., & Bernhardsson, L. (2018). Action design research in practice: lessons and concerns.
- Helveston, J. P., Feit, E. M., & Michalek, J. J. (2018). Pooling stated and revealed preference data in the presence of rp endogeneity. *Transportation Research Part B: Methodological*, *109*, 70–89.
- Hensher, D. A., Rose, J., & Greene, W. H. (2005). The implications on willingness to pay of respondents ignoring specific attributes. *Transportation*, *32*(3), 203–222.
- Hevner, A., & Chatterjee, S. (2010). Design science research in information systems. In *Design research in information systems* (pp. 9–22). Springer.

- Hevner, A. R. (2007). A three cycle view of design science research. *Scandinavian journal of information systems*, 19(2), 4.
- Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. *MIS quarterly*, 75–105.
- Hoepman, J.-H. (2014). Privacy design strategies. In *Ifip international information security conference* (pp. 446–459).
- Holst, H., Åström, K., Järund, A., Palmer, J., Heyden, A., Kahl, F., ... Edenbrandt, L. (2000). Automated interpretation of ventilation-perfusion lung scintigrams for the diagnosis of pulmonary embolism using artificial neural networks. *European journal of nuclear medicine*, 27(4), 400–406.
- Holzinger, A., Biemann, C., Pattichis, C. S., & Kell, D. B. (2017). What do we need to build explainable ai systems for the medical domain? *arXiv preprint arXiv:1712.09923*.
- Hong, T.-P., Wang, C.-Y., & Lin, C.-W. (2010). Providing timely updated sequential patterns in decision making. *International Journal of Information Technology & Decision Making*, 9(06), 873–888.
- Hoogervorst, J. A. (2009). *Enterprise governance and enterprise engineering*. Springer Science & Business Media.
- Horsky, J., Schiff, G. D., Johnston, D., Mercincavage, L., Bell, D., & Middleton, B. (2012). Interface design principles for usable decision support: a targeted review of best practices for clinical prescribing interventions. *Journal of biomedical informatics*, 45(6), 1202–1216.
- Huang, Y.-M., Hung, C.-M., & Jiau, H. C. (2006). Evaluation of neural networks and data mining methods on a credit assessment task for class imbalance problem. *Nonlinear Analysis: Real World Applications*, 7(4), 720–747.
- Huber, J., & Zwerina, K. (1996). The importance of utility balance in efficient choice designs. *Journal of Marketing research*, 33(3), 307–317.
- Hussain, M., Afzal, M., Khan, W. A., & Lee, S. (2012). Clinical decision support service for elderly people in smart home environment. In *2012 12th international conference on control automation robotics & vision (icarcv)* (pp. 678–683).
- Hutton, R. J., & Klein, G. (1999). Expert decision making. *Systems Engineering: The Journal of The International Council on Systems Engineering*, 2(1), 32–45.
- Hyysalo, S., & Lehenkari, J. (2003). An activity-theoretical method for studying user participation in is design. *Methods of Information in Medicine*, 42(04), 398–404.
- Ibrahim, A. M., & Bennett, B. (2014). The assessment of machine learning model performance for predicting alluvial deposits distribution. *Procedia Computer Science*, 36, 637–642.
- Iivari, J. (2015). Distinguishing and contrasting two strategies for design science research. *European Journal of Information Systems*, 24(1), 107–115.
- Islam, R., Weir, C., & Del Fiore, G. (2014). Heuristics in managing complex clinical decision tasks in experts' decision making. In *2014 IEEE International Conference on Healthcare Informatics* (pp. 186–193).
- Johannesson, P., & Perjons, E. (2014). *An introduction to design science*. Springer.
- Jordan, M. I., & Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245), 255–260.
- Jung, Y. (2018). Multiple predicting k-fold cross-validation for model selection. *Journal of Nonparametric Statistics*, 30(1), 197–215.
- Kawamoto, K., Houlihan, C. A., Balas, E. A., & Lobach, D. F. (2005). Improving clinical practice using clinical decision support systems: a systematic review of trials to identify features critical to success. *Bmj*, 330(7494), 765.
- Kazman, R., Bass, L., Abowd, G., & Webb, M. (1994). Saam: A method for analyzing the properties of software architectures. In *Proceedings of 16th international conference on software engineering* (pp. 81–90).
- Kazman, R., Klein, M., Barbacci, M., Longstaff, T., Lipson, H., & Carriere, J. (1998). The architecture tradeoff analysis method. In *Proceedings. fourth IEEE international conference on engineering of complex computer systems (cat. no. 98ex193)* (pp. 68–78).
- Keijzer-Broers, W. (2016). Developing a service platform for health and wellbeing in a living lab setting: an action design research approach.
- Keijzer-Broers, W., & de Reuver, M. (2016). Applying agile design sprint methods in action design research: prototyping a health and wellbeing platform. In *International conference on design science research in information system and technology* (pp. 68–80).
- Keil, M., Beranek, P. M., & Konsynski, B. R. (1995). Usefulness and ease of use: field study evidence regarding task considerations. *Decision support systems*, 13(1), 75–91.

- Keys, Y., Silverman, S. R., & Evans, J. (2017). Identification of tools and techniques to enhance interdisciplinary collaboration during design and construction projects. *HERD: Health Environments Research & Design Journal*, *10*(5), 28–38.
- Khairat, S., Marc, D., Crosby, W., & Al Sanousi, A. (2018). Reasons for physicians not adopting clinical decision support systems: critical analysis. *JMIR medical informatics*, *6*(2), e24.
- Khalifa, M. (2014). Clinical decision support: Strategies for success. *Procedia Computer Science*, *37*, 422–427.
- Kim, J. A., Cho, I., & Kim, Y. (2008). Cdss (clinical decision support system) architecture in korea. In *2008 international conference on convergence and hybrid information technology* (pp. 700–703).
- Kirkebøen, G. (2009). Decision behaviour-improving expert judgement. In *Making essential choices with scant information* (pp. 169–194). Springer.
- Klein Koerkamp, R. (2019). *The road from analytical cdss invention to implementation in health-care* (Unpublished master’s thesis). University of Twente.
- Klinkenberg, R., & Rüping, S. (2002). Concept drift and the importance of examples. In *Text mining—theoretical aspects and applications*.
- Knodel, J., Muthig, D., & Naab, M. (2006). Static architecture evaluation of open source reuse candidates. *NODe 2006–GSEM 2006*.
- Kolter, J. Z., & Maloof, M. A. (2007). Dynamic weighted majority: An ensemble method for drifting concepts. *The Journal of Machine Learning Research*, *8*, 2755–2790.
- Kong, G., Xu, D.-L., & Yang, J.-B. (2008). Clinical decision support systems: a review on knowledge representation and inference under uncertainties. *International Journal of Computational Intelligence Systems*, *1*(2), 159–167.
- Korva, N., Porter, S., O’Connor, B. P., Shaw, J., & Brinke, L. t. (2013). Dangerous decisions: Influence of juror attitudes and defendant appearance on legal decision-making. *Psychiatry, Psychology and Law*, *20*(3), 384–398.
- Kotsiantis, S. B., Zaharakis, I. D., & Pintelas, P. E. (2006). Machine learning: a review of classification and combining techniques. *Artificial Intelligence Review*, *26*(3), 159–190.
- Kruse, C. S., Goswamy, R., Raval, Y. J., & Marawi, S. (2016). Challenges and opportunities of big data in health care: a systematic review. *JMIR medical informatics*, *4*(4), e38.
- Kuechler, B., & Vaishnavi, V. (2008). On theory development in design science research: anatomy of a research project. *European Journal of Information Systems*, *17*(5), 489–504.
- Kumar, A. (2015). Stakeholder’s perspective of clinical decision support system. *Open Journal of Business and Management*, *4*(1), 45–50.
- Kusumasondjaja, S., Shanka, T., & Marchegiani, C. (2012). Credibility of online reviews and initial trust: The roles of reviewer’s identity and review valence. *Journal of Vacation Marketing*, *18*(3), 185–195.
- Lachaab, M., Ansari, A., Jedidi, K., & Trabelsi, A. (2006). Modeling preference evolution in discrete choice models: A bayesian state-space approach. *Quantitative Marketing and Economics*, *4*(1), 57–81.
- Lajnef, M. A., Ayed, M. B., & Kolski, C. (2005). Convergence possible des processus du data mining et de conception-évaluation d’ihm: adaptation du modèle en u. In *Proceedings of the 17th conference on l’interaction homme-machine* (pp. 243–246).
- Lakshmanaprabu, S., Mohanty, S. N., Krishnamoorthy, S., Uthayakumar, J., Shankar, K., et al. (2019). Online clinical decision support system using optimal deep neural networks. *Applied Soft Computing*, *81*, 105487.
- Lassing, N., Bengtsson, P., Van Vliet, H., & Bosch, J. (2002). Experiences with alma: architecture-level modifiability analysis. *Journal of systems and software*, *61*(1), 47–57.
- Lavasani, M., Hossan, M. S., Asgari, H., & Jin, X. (2017). Examining methodological issues on combined rp and sp data. *Transportation research procedia*, *25*, 2330–2343.
- Lee, C., Ran, B., Yang, F., & Loh, W.-Y. (2010). A hybrid tree approach to modeling alternate route choice behavior with online information. *Journal of Intelligent Transportation Systems*, *14*(4), 209–219.
- Lee, K.-W., & Huh, S.-Y. (2006). A model-solver integration framework for autonomous and intelligent model solution. *Decision Support Systems*, *42*(2), 926–944.
- Leist, S., & Zellner, G. (2006). Evaluation of current architecture frameworks. In *Proceedings of the 2006 acm symposium on applied computing* (pp. 1546–1553).
- Lenz, A. (2020). Designing dynamic decision support for electronic requirements negotiations. In *Dynamic decision support for electronic requirements negotiations* (pp. 75–89). Springer.
- Lesko, C. R., Buchanan, A. L., Westreich, D., Edwards, J. K., Hudgens, M. G., & Cole, S. R. (2017). Generalizing study results: a potential outcomes perspective. *Epidemiology (Cambridge)*,

- Mass.*), 28(4), 553.
- Li, X., Hess, T. J., & Valacich, J. S. (2008). Why do we trust new technology? a study of initial trust formation with organizational information systems. *The Journal of Strategic Information Systems*, 17(1), 39–71.
- Liberati, E. G., Ruggiero, F., Galuppo, L., Gorli, M., González-Lorenzo, M., Maraldi, M., ... others (2017). What hinders the uptake of computerized decision support systems in hospitals? a qualitative study and framework for implementation. *Implementation Science*, 12(1), 1–13.
- Lindstrom, A. (2006). On the syntax and semantics of architectural principles. In *Proceedings of the 39th annual hawaii international conference on system sciences (hicss'06)* (Vol. 8, pp. 178b–178b).
- Longhurst, R. (2003). Semi-structured interviews and focus groups. *Key methods in geography*, 3(2), 143–156.
- Louviere, J., & Timmermans, H. (1990). Stated preference and choice models applied to recreation research: a review. *Leisure Sciences*, 12(1), 9–32.
- Louviere, J. J., Flynn, T. N., & Carson, R. T. (2010). Discrete choice experiments are not conjoint analysis. *Journal of choice modelling*, 3(3), 57–72.
- Louviere, J. J., Hensher, D. A., & Swait, J. D. (2000). *Stated choice methods: analysis and applications*. Cambridge university press.
- Louviere, J. J., Meyer, R. J., Bunch, D. S., Carson, R., Dellaert, B., Hanemann, W. M., ... Irwin, J. (1999). Combining sources of preference data for modeling complex decision processes. *Marketing Letters*, 10(3), 205–217.
- Ltifi, H., Ayed, M. B., Kolski, C., & Alimi, A. M. (2009). Hci-enriched approach for dss development: the up/u approach. In *2009 ieee symposium on computers and communications* (pp. 895–900).
- Lu, X. (2005). An investigation on service-oriented architecture for constructing distributed web gis application. In *2005 ieee international conference on services computing (scc'05) vol-1* (Vol. 1, pp. 191–197).
- Lyman, J. A., Cohn, W. F., Bloomrosen, M., & Detmer, D. E. (2010). Clinical decision support: progress and opportunities. *Journal of the American Medical Informatics Association*, 17(5), 487–492.
- Maccani, G., Donnellan, B., & Helfert, M. (2014). Action design research in practice: the case of smart cities. In *International conference on design science research in information systems* (pp. 132–147).
- Mackworth, N. H., et al. (1950). Researches on the measurement of human performance. *Researches on the Measurement of Human Performance*.(268).
- Madura, J. (2006). *Introduction to business*. Cengage Learning.
- Magee, J., Dulay, N., Eisenbach, S., & Kramer, J. (1995). Specifying distributed software architectures. In *European software engineering conference* (pp. 137–153).
- Mahiddin, N., Othman, Z., & Bakar, A. (2017). An architecture of multiagent system (mas) for healthcare intelligent decision support system (idss). *Journal of Fundamental and Applied Sciences*, 9(5S), 144–167.
- Maiden, N. A., & Hare, M. (1998). Problem domain categories in requirements engineering. *International Journal of Human-Computer Studies*, 49(3), 281–304.
- March, S. T., & Smith, G. F. (1995). Design and natural science research on information technology. *Decision support systems*, 15(4), 251–266.
- Mårtensson, F. (2006). *Software architecture quality evaluation: Approaches in an industrial context* (Unpublished doctoral dissertation). Blekinge Institute of Technology.
- Martensson, F., Grahn, H., & Mattsson, M. (2004). Prototype-based software architecture evaluation—component quality attribute evaluation. In *Proceedings of the 4th conference on software engineering research and practice in sweden* (pp. 11–17).
- McIntosh, M. J., & Morse, J. M. (2015). Situating and constructing diversity in semi-structured interviews. *Global qualitative nursing research*, 2, 2333393615597674.
- Meertens, L. O., Iacob, M.-E., & Nieuwenhuis, L. J. (2010). Goal and model driven design of an architecture for a care service platform. In *Proceedings of the 2010 acm symposium on applied computing* (pp. 158–164).
- Meister, D., & Enderwick, T. P. (2001). *Human factors in system design, development, and testing*. CRC Press.
- Melton, B. L., Zillich, A. J., Saleem, J. J., Russ, A. L., Tisdale, J. E., & Overholser, B. R. (2016). Iterative development and evaluation of a pharmacogenomic-guided clinical decision support system for warfarin dosing. *Applied clinical informatics*, 7(4), 1088.

- Miah, S. J., Blake, J., & Kerr, D. (2020). Meta-design knowledge for clinical decision support systems. *Australasian Journal of Information Systems*, 24, 1–21.
- Michalewicz, Z., Schmidt, M., Michalewicz, M., & Chiriach, C. (2006). *Adaptive business intelligence*. Springer.
- Montani, S., & Striani, M. (2019). Artificial intelligence in clinical decision support: a focused literature survey. *Yearbook of medical informatics*, 28(1), 120.
- Moore, J. H., & Chang, M. G. (1980). Design of decision support systems. *ACM SIGOA Newsletter*, 1(4-5), 8–14.
- Morikawa, T. (1989). *Incorporating stated preference data in travel demand analysis* (Unpublished doctoral dissertation). Massachusetts Institute of Technology.
- Nielsen, J., & Molich, R. (1990). Heuristic evaluation of user interfaces. In *Proceedings of the sigchi conference on human factors in computing systems* (pp. 249–256).
- O’Brien, L., Merson, P., & Bass, L. (2007). Quality attributes for service-oriented architectures. In *International workshop on systems development in soa environments (sdsoa’07: Icse workshops 2007)* (pp. 3–3).
- Oh, S., Cha, J., Ji, M., Kang, H., Kim, S., Heo, E., ... others (2015). Architecture design of healthcare software-as-a-service platform for cloud-based clinical decision support service. *Healthcare informatics research*, 21(2), 102.
- Oreizy, P., Gorlick, M. M., Taylor, R. N., Heimhigner, D., Johnson, G., Medvidovic, N., ... Wolf, A. L. (1999). An architecture-based approach to self-adaptive software. *IEEE Intelligent Systems and Their Applications*, 14(3), 54–62.
- Osop, H., & Sahama, T. (2019). Systems design framework for a practice-based evidence approached clinical decision support systems. In *Proceedings of the australasian computer science week multiconference* (pp. 1–6).
- Parasuraman, R., & Riley, V. (1997). Humans and automation: Use, misuse, disuse, abuse. *Human factors*, 39(2), 230–253.
- Patidar, A., & Suman, U. (2015). A survey on software architecture evaluation methods. In *2015 2nd international conference on computing for sustainable global development (indiacom)* (pp. 967–972).
- Peppers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of management information systems*, 24(3), 45–77.
- Pérez, I. J., Cabrerizo, F. J., & Herrera-Viedma, E. (2010). A mobile decision support system for dynamic group decision-making problems. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 40(6), 1244–1256.
- Perry, D. E., & Wolf, A. L. (1992). Foundations for the study of software architecture. *ACM SIGSOFT Software engineering notes*, 17(4), 40–52.
- Petersson, A. M., & Lundberg, J. (2016). Applying action design research (adr) to develop concept generation and selection methods. *Procedia Cirp*, 50, 222–227.
- Pirnejad, H., Amiri, P., Niazkhani, Z., Shiva, A., Makhdoomi, K., Abkhiz, S., ... Bal, R. (2019). Preventing potential drug-drug interactions through alerting decision support systems: a clinical context based methodology. *International journal of medical informatics*, 127, 18–26.
- Pitts, M. G., & Browne, G. J. (2007). Improving requirements elicitation: an empirical investigation of procedural prompts. *Information systems journal*, 17(1), 89–110.
- Pombo Jimenez, D. (2017). Design of a flexible ict architecture for the integration of floating car data in rijkswaterstaat’s traffic management and information systems: A design science research approach.
- Power, D. J. (2002). *Decision support systems: concepts and resources for managers*. Greenwood Publishing Group.
- Power, D. J. (2008). Decision support systems: a historical overview. In *Handbook on decision support systems 1* (pp. 121–140). Springer.
- Prezenski, S., Brechmann, A., Wolff, S., & Russwinkel, N. (2017). A cognitive modeling approach to strategy formation in dynamic decision making. *Frontiers in psychology*, 8, 1335.
- Proper, E., & Greefhorst, D. (2010). The roles of principles in enterprise architecture. In *International workshop on trends in enterprise architecture research* (pp. 57–70).
- Qiao, Y., Huang, Y., Yang, F., Zhang, M., & Chen, L. (2016). Empirical study of travel mode forecasting improvement for the combined revealed preference/stated preference data-based discrete choice model. *Advances in Mechanical Engineering*, 8(1), 1687814015624836.
- Queirós, A., Faria, D., & Almeida, F. (2017). Strengths and limitations of qualitative and quantitative research methods. *European Journal of Education Studies*.

- Raghupathi, W., & Raghupathi, V. (2014). Big data analytics in healthcare: promise and potential. *Health information science and systems*, 2(1), 1–10.
- Rajer-Kanduč, K., Zupan, J., & Majcen, N. (2003). Separation of data on the training and test set for modelling: a case study for modelling of five colour properties of a white pigment. *Chemometrics and intelligent laboratory systems*, 65(2), 221–229.
- Raschka, S. (2018). Model evaluation, model selection, and algorithm selection in machine learning. *arXiv preprint arXiv:1811.12808*.
- Ravikumar, K., MacLaughlin, K. L., Scheitel, M. R., Kessler, M., Waghlikar, K. B., Liu, H., & Chaudhry, R. (2018). Improving the accuracy of a clinical decision support system for cervical cancer screening and surveillance. *Applied clinical informatics*, 9(1), 62.
- Rawson, T., Moore, L., Hernandez, B., Charani, E., Castro-Sanchez, E., Herrero, P., ... Holmes, A. (2017). A systematic review of clinical decision support systems for antimicrobial management: are we failing to investigate these interventions appropriately? *Clinical Microbiology and Infection*, 23(8), 524–532.
- Rizzi, L. I., & de Dios Ortúzar, J. (2003). Stated preference in the valuation of interurban road safety. *Accident Analysis & Prevention*, 35(1), 9–22.
- Rogers, E. M., & Shoemaker, F. F. (1971). Communication of innovations; a cross-cultural approach.
- Rosenberger, R. S., Peterson, G. L., Clarke, A., & Brown, T. C. (2003). Measuring dispositions for lexicographic preferences of environmental goods: integrating economics, psychology and ethics. *Ecological Economics*, 44(1), 63–76.
- Røst, T. B., Clausen, C., Nytrø, Ø., Kuposov, R., Leventhal, B., Westbye, O. S., ... Skokauskas, N. (2020). Local, early, and precise: Designing a clinical decision support system for child and adolescent mental health services. *Frontiers in Psychiatry*, 11, 1473.
- Rouhani, B. D., Mahrin, M. N., Nikpay, F., Ahmad, R. B., & Nikfard, P. (2015). A systematic literature review on enterprise architecture implementation methodologies. *information and Software Technology*, 62, 1–20.
- Rozanski, N., & Woods, E. (2012). *Software systems architecture: working with stakeholders using viewpoints and perspectives*. Addison-Wesley.
- Rubinstein, I. S., & Good, N. (2013). Privacy by design: A counterfactual analysis of google and facebook privacy incidents. *Berkeley Tech. LJ*, 28, 1333.
- Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5), 206–215.
- Sambasivan, M., Esmailzadeh, P., Kumar, N., & Nezakati, H. (2012). Intention to adopt clinical decision support systems in a developing country: effect of physician’s perceived professional autonomy, involvement and belief: a cross-sectional study. *BMC medical informatics and decision making*, 12(1), 1–8.
- Sanchez, E., Toro, C., Carrasco, E., Bueno, G., Parra, C., Bonachela, P., ... Guijarro, F. (2011). An architecture for the semantic enhancement of clinical decision support systems. In *International conference on knowledge-based and intelligent information and engineering systems* (pp. 611–620).
- Sanko, N. (2001). Guidelines for stated preference experiment design. *Master of Business Administration diss., Ecole Nationale des Ponts et Chaussées. s*.
- Saqlain, S. M., Sher, M., Shah, F. A., Khan, I., Ashraf, M. U., Awais, M., & Ghani, A. (2019). Fisher score and matthews correlation coefficient-based feature subset selection for heart disease diagnosis using support vector machines. *Knowledge and Information Systems*, 58(1), 139–167.
- Savolainen, J., & Myllarniemi, V. (2009). Layered architecture revisited—comparison of research and practice. In *2009 joint working ieee/ifip conference on software architecture & european conference on software architecture* (pp. 317–320).
- Sein, M. K., Henfridsson, O., Purao, S., Rossi, M., & Lindgren, R. (2011). Action design research. *MIS quarterly*, 37–56.
- Shah, T., & Patel, S. (2016). A novel approach for specifying functional and non-functional requirements using rds (requirement description schema). *Procedia computer science*, 79, 852–860.
- Shahin, M., Liang, P., & Khayyambashi, M. R. (2010). Improving understandability of architecture design through visualization of architectural design decision. In *Proceedings of the 2010 icse workshop on sharing and reusing architectural knowledge* (pp. 88–95).
- Shaikh, F., Dehmeshki, J., Bisdas, S., Roettger-Dupont, D., Kubassova, O., Aziz, M., & Awan, O. (2020). Artificial intelligence-based clinical decision support systems using advanced medical imaging & radiomics. *Current Problems in Diagnostic Radiology*.

- Shanmugapriya, P., & Suresh, R. (2012). Software architecture evaluation methods-a survey. *International Journal of Computer Applications*, 49(16).
- Shaw, M., & Garlan, D. (1996). Software architecture: Perspectives on an engineering discipline.
- Shim, J. P., Warkentin, M., Courtney, J. F., Power, D. J., Sharda, R., & Carlsson, C. (2002). Past, present, and future of decision support technology. *Decision support systems*, 33(2), 111–126.
- Siau, K., & Shen, Z. (2003). Building customer trust in mobile commerce. *Communications of the ACM*, 46(4), 91–94.
- Siau, K., & Wang, W. (2018). Building trust in artificial intelligence, machine learning, and robotics. *Cutter Business Technology Journal*, 31(2), 47–53.
- Siddarth, S., Bucklin, R. E., & Morrison, D. G. (1995). Making the cut: Modeling and analyzing choice set restriction in scanner panel data. *Journal of Marketing Research*, 32(3), 255–266.
- Sidiropoulos, K., Glotsos, D., Kostopoulos, S., Ravazoula, P., Kalatzis, I., Cavouras, D., & Stoham, J. (2012). Real time decision support system for diagnosis of rare cancers, trained in parallel, on a graphics processing unit. *Computers in biology and medicine*, 42(4), 376–386.
- Silver, M. S. (1991). Decisional guidance for computer-based decision support. *MIS quarterly*, 105–122.
- Sim, I., Gorman, P., Greenes, R. A., Haynes, R. B., Kaplan, B., Lehmann, H., & Tang, P. C. (2001). Clinical decision support systems for the practice of evidence-based medicine. *Journal of the American Medical Informatics Association*, 8(6), 527–534.
- Sinha, R., & Swearingen, K. (2002). The role of transparency in recommender systems. In *Chi'02 extended abstracts on human factors in computing systems* (pp. 830–831).
- Smith, P. J., Geddes, N. D., & Beatty, R. (2009). Human-centered design of decision-support systems. *Human-Computer Interaction: Design Issues, Solutions, and Applications*, 245.
- Song, X., Danaf, M., Atasoy, B., & Ben-Akiva, M. (2018). Personalized menu optimization with preference updater: a boston case study. *Transportation Research Record*, 2672(8), 599–607.
- Stafford, J. A., Richardson, D. J., & Wolf, A. L. (1998). *Aladdin: A tool for architecture-level dependence analysis of software systems* (Tech. Rep.). COLORADO UNIV AT BOULDER DEPT OF COMPUTER SCIENCE.
- Steinberg, D., & Scott Cardell, N. (1992). Estimating logistic regression models when the dependent variable has no variance. *Communications in Statistics-Theory and Methods*, 21(2), 423–450.
- Stevanetic, S., & Zdun, U. (2015). Software metrics for measuring the understandability of architectural structures: a systematic mapping study. In *Proceedings of the 19th international conference on evaluation and assessment in software engineering* (pp. 1–14).
- Sutton, R. T., Pincock, D., Baumgart, D. C., Sadowski, D. C., Fedorak, R. N., & Kroeker, K. I. (2020). An overview of clinical decision support systems: benefits, risks, and strategies for success. *NPJ digital medicine*, 3(1), 1–10.
- Swait, J., & Louviere, J. (1993). The role of the scale parameter in the estimation and comparison of multinomial logit models. *Journal of marketing research*, 30(3), 305–314.
- Swait, J., Louviere, J. J., & Williams, M. (1994). A sequential approach to exploiting the combined strengths of sp and rp data: application to freight shipper choice. *Transportation*, 21(2), 135–152.
- Tabares, F., Hernandez, J., & Cabezas, I. (2017). Architectural design of a clinical decision support system for clinical triage in emergency departments. In *Colombian conference on computing* (pp. 267–281).
- Tabares, L., Hernandez, J., & Cabezas, I. (2016). Architectural approaches for implementing clinical decision support systems in cloud: a systematic review. In *2016 IEEE First International Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE)* (pp. 42–47).
- Tariq, A., & Rafi, K. (2012). Intelligent decision support systems-a framework. In *Information and knowledge management* (Vol. 2, pp. 12–20).
- ten Broeke, A. (2020). A new approach to artificial intelligence for decision support: Case study in the neonatal intensive care unit of the university medical centre of groningen.
- Ten Broeke, A., Hulscher, J., Heyning, N., Kooi, E., & Chorus, C. (2021). Bait: A new medical decision support technology based on discrete choice theory. *Medical Decision Making*, 0272989X211001320.
- Tharwat, A. (2020). Classification assessment methods. *Applied Computing and Informatics*.
- Tomaszewski, W. (2012). Computer-based medical decision support system based on guidelines, clinical pathways and decision nodes. *Acta of Bioengineering & Biomechanics*, 14(1).
- Train, K. E. (2009). *Discrete choice methods with simulation*. Cambridge university press.

- Trivedi, M. H., Daly, E. J., Kern, J. K., Grannemann, B. D., Sunderajan, P., & Claassen, C. A. (2009). Barriers to implementation of a computerized decision support system for depression: an observational report on lessons learned in” real world” clinical settings. *BMC medical informatics and decision making*, *9*(1), 1–9.
- Vahidov, R., & Kersten, G. E. (2004). Decision station: situating decision support systems. *Decision Support Systems*, *38*(2), 283–303.
- Vaishnavi, V., & Kuechler, W. (2004). Design research in information systems.
- Vaishnavi, V. K., & Kuechler, W. (2015). *Design science research methods and patterns: innovating information and communication technology*. Crc Press.
- Van Cranenburgh, S., Wang, S., Vij, A., Pereira, F., & Walker, J. (2021). Choice modelling in the age of machine learning. *arXiv preprint arXiv:2101.11948*.
- Varonen, H., Kortteisto, T., Kaila, M., & Group, E. S. (2008). What may help or hinder the implementation of computerized decision support systems (cdsss): a focus group study with physicians. *Family practice*, *25*(3), 162–167.
- Velickovski, F., Ceccaroni, L., Roca, J., Burgos, F., Galdiz, J. B., Marina, N., & Lluch-Ariet, M. (2014). Clinical decision support systems (cdss) for preventive management of copd patients. *Journal of translational medicine*, *12*(2), 1–10.
- Venable, J., Pries-Heje, J., & Baskerville, R. (2012). A comprehensive framework for evaluation in design science research. In *International conference on design science research in information systems* (pp. 423–438).
- Venkatesh, V. (2000). Determinants of perceived ease of use: Integrating control, intrinsic motivation, and emotion into the technology acceptance model. *Information systems research*, *11*(4), 342–365.
- Verschuren, P., Doorewaard, H., & Mellion, M. (2010). *Designing a research project* (Vol. 2). Eleven International Publishing The Hague.
- Verschuren, P., & Hartog, R. (2005). Evaluation in design-oriented research. *Quality and Quantity*, *39*(6), 733–762.
- Vogel, O., Arnold, I., Chughtai, A., & Kehrer, T. (2011). *Software architecture: a comprehensive framework and guide for practitioners*. Springer Science & Business Media.
- Wagholikar, K. B., MacLaughlin, K. L., Kastner, T. M., Casey, P. M., Henry, M., Greenes, R. A., ... Chaudhry, R. (2013). Formative evaluation of the accuracy of a clinical decision support system for cervical cancer screening. *Journal of the American Medical Informatics Association*, *20*(4), 749–757.
- Wagholikar, K. B., Sundararajan, V., & Deshpande, A. W. (2012). Modeling paradigms for medical diagnostic decision support: a survey and future directions. *Journal of medical systems*, *36*(5), 3029–3049.
- Wang, D., Wang, L., Zhang, Z., Wang, D., Zhu, H., Gao, Y., ... Tian, F. (2021). ” brilliant ai doctor” in rural china: Tensions and challenges in ai-powered cdss deployment. *arXiv preprint arXiv:2101.01524*.
- Washington, S., Ravulaparthi, S., Rose, J. M., Hensher, D., & Pendyala, R. (2014). Bayesian imputation of non-chosen attribute values in revealed preference surveys. *Journal of Advanced Transportation*, *48*(1), 48–65.
- Wilk, S., Michalowski, W., O’Sullivan, D., Farion, K., Sayyad-Shirabad, J., Kuziemy, C., & Kukawka, B. (2013). A task-based support architecture for developing point-of-care clinical decision support systems for the emergency department. *Methods of information in medicine*, *52*(1), 18–32.
- Winter, R., & Aier, S. (2011). How are enterprise architecture design principles used? In *2011 ieee 15th international enterprise distributed object computing conference workshops* (pp. 314–321).
- Wong, T.-T. (2015). Performance evaluation of classification algorithms by k-fold and leave-one-out cross validation. *Pattern Recognition*, *48*(9), 2839–2846.
- Wu, G., Yang, P., Xie, Y., Woodruff, H. C., Rao, X., Guiot, J., ... others (2020). Development of a clinical decision support system for severity risk prediction and triage of covid-19 patients at hospital admission: an international multicentre study. *European Respiratory Journal*, *56*(2).
- Xafis, V., Schaefer, G. O., Labude, M. K., Brassington, I., Ballantyne, A., Lim, H. Y., ... others (2019). An ethics framework for big data in health and research. *Asian Bioethics Review*, *11*(3), 227–254.
- Xiao, L., Cousins, G., Fahey, T., Dimitrov, B. D., & Hederman, L. (2012). Developing a rule-driven clinical decision support system with an extensive and adaptative architecture. In *2012 ieee*

- 14th international conference on e-health networking, applications and services (healthcom) (pp. 250–254).
- Xiong, Z., Cui, Y., Liu, Z., Zhao, Y., Hu, M., & Hu, J. (2020). Evaluating explorative prediction power of machine learning algorithms for materials discovery using k-fold forward cross-validation. *Computational Materials Science*, *171*, 109203.
- Xu, Y., & Goodacre, R. (2018). On splitting training and validation set: A comparative study of cross-validation, bootstrap and systematic sampling for estimating the generalization performance of supervised learning. *Journal of Analysis and Testing*, *2*(3), 249–262.
- Yan, C. (2018). *Developing digital support for learning and diagnostic reasoning in clinical practice* (Unpublished doctoral dissertation). Umeå University.
- Yao, W., & Kumar, A. (2013). Conflexflow: integrating flexible clinical pathways into clinical decision support systems using context and rules. *Decision Support Systems*, *55*(2), 499–515.
- Yeung, A. K., & Hall, G. B. (2007). *Spatial database systems: design, implementation and project management* (Vol. 87). Springer Science & Business Media.
- Yilmaz, L., & Tolk, A. (2008). A unifying multimodel taxonomy and agent-supported multisimulation strategy for decision-support. In *Intelligent decision making: An ai-based approach* (pp. 193–226). Springer.
- Yu, H., Breslau, L., & Shenker, S. (1999). A scalable web cache consistency architecture. *ACM SIGCOMM Computer Communication Review*, *29*(4), 163–174.
- Yun, Y., Ma, D., & Yang, M. (2021). Human–computer interaction-based decision support system with applications in data mining. *Future Generation Computer Systems*, *114*, 285–289.
- Zachary, W. W. (1988). Decision support systems: Designing to extend the cognitive limits. In *Handbook of human-computer interaction* (pp. 997–1030). Elsevier.
- Zeleznikow, J., & Nolan, J. R. (2001). Using soft computing to build real world intelligent decision support systems in uncertain domains. *Decision Support Systems*, *31*(2), 263–285.
- Zhang, G., Xu, Y., & Li, T. (2012). Guest editorial: a special issue on new trends in intelligent decision support systems. *Knowledge-Based Systems*, *32*, 1–2.
- Zhang, Y.-F., Gou, L., Tian, Y., Li, T.-C., Zhang, M., & Li, J.-S. (2016). Design and development of a sharable clinical decision support system based on a semantic web service framework. *Journal of medical systems*, *40*(5), 118.
- Zhou, J. Y. (2004). *Functional requirements and non-functional requirements: a survey* (Unpublished doctoral dissertation). Concordia University.
- Zhu, X., Feng, J., Huang, S., & Chen, C. (2020). An online updating method for time-varying preference learning. *Transportation Research Part C: Emerging Technologies*, *121*, 102849.
- Zikos, D., & DeLellis, N. (2018). Cdss-rm: a clinical decision support system reference model. *BMC medical research methodology*, *18*(1), 1–14.

Appendix A

Overview of the interviews

Table A.1: *List of the requirement identification interviews*

Reference	Date	Respondent
Requirement interview 1 Clinical end user	5 January 2021, 9:00am - 10:00am	Jan Hulscher, UMCG
Requirement interview 2 Clinical end user	5 January 2021, 2:00pm - 3:00pm	Jean-Paul de Vries, UMCG
Requirement interview 3 Clinical end user	12 January 2021, 5:00pm - 6:00pm	Lucas Savalle, Medical Centre Haaglanden
Requirement interview 1 Council	15 January 2021, 10:00am - 11:00am	Nicolaas Heyning
Requirement interview 2 Council	15 January 2021, 3:30pm - 4:00pm	Caspar Chorus
Requirement interview 3 Council	14 January 2021, 1:30pm - 2:30pm	Annebel ten Broeke

Appendix B

Interview analysis

This appendix presents an analysis of the interviews. The goal of this analysis is to identify architecture requirements that can inform the design of the architecture. The interviews were conducted with stakeholders that are involved in the design at two different levels. The stakeholders interviewed are three representatives of Councilyl and three clinical end users.

The representatives of Councilyl are:

1. Co-founder and CEO of Councilyl: has a high stake in the design of the architecture, since this participant has to use the architecture in future to help different healthcare clients with decision-making in dynamic contexts. This participant helps the clients to codify, support and automate decision processes, and will implement the full architecture in the existing software.
2. Co-founder and scientific advisor of Councilyl: has a high influence, because of the knowledge about the science of discrete choice modeling.
3. Decision analyst of Councilyl: has a high stake in the design of the architecture, since this participant will use components defined in the architecture to apply a BAIT-based CDSS in various healthcare decision-making contexts

The clinical end users interviewed are:

1. One clinical end user that is considered an “advanced” client of Councilyl. The choice tasks occurs rarely: approximately 20 times a year.
2. One clinical end user that is considered an “new” client of Councilyl. The decision-making tasks occurs frequently: approximately 100 times a year (bi-weekly).
3. One clinical end user that is considered not a client of Councilyl. The choice tasks occurs relatively frequent: approximately 50 times a year (weekly).

The findings of the interviews were translated into requirements at a single level: the level of the architecture. This was done by consulting literature review as well for three reasons. First of all, the interviews are about an extension of a CDSS that already exists. As a consequence, respondents may only discuss what is not in place yet. However, some aspects that are covered need to be addressed by the architecture of the desired CDSS as well. An example is reliability. Second, the interviewed clinical end users form just a small selection which may result in somewhat biased answers. Third, interviews are subjective by nature. Literature provides a more general and objective input on the requirement identification. Below, the analysis of the interviews with both Councilyl (see appendix B.1) and the three clinical end users (see appendix B.2) are presented.

B.1 Interviews Councilyl

B.1.1 Interview part 1: Main purpose

Two of the representatives (2 and 3), stated that the architecture should enable Councilyl to confidently tell clients they can serve them with a dynamic support tool that can be developed in a foreseeable time, since Councilyl has the groundwork –the architecture – already in place. This goal can be further broken down.

First of all, this implies that the groundwork must be in a state such it allows Council to directly develop CDSS given the preferences of the clinical end users. As such, Council should be able to trust the architecture and doubtlessly follow what the architecture defines. First of all, it is important that the architecture is reliable with regard to what the architecture describes. Accordingly, the processes in the architecture should be tested. The test should determine if the processes in the architecture effectively work for the goal of the process. Moreover, it is thereby important that the developers follow the architecture as intended. This means that the architecture can only be interpreted in one way and should therefore explicate all aspects that are relevant for developers, but might not be directly clear by only presenting CDSS components.

- The architecture should define the model update engine only with processes that have proved to achieve the goal for which the architecture includes the processes.
- The architecture should mark parallel processes that have to run at the same time.
- The architecture should mark processes that require collaboration between a clinical end user and a CDSS.
- The architecture should be designed according to one description language.
- The architecture should distinguish components that produce information and components that consume information.

Representative 2 explains that the architecture should generate a well-functioning prototype given preferences of the client. This prototype should be developed and implemented in foreseeable time. In the context of Council, this can be further specified. According to representative 1 and 2, the current CDSS is developed and implemented within a month. Adding the components that make the CDSS dynamic, may take another three to four weeks.

- The architecture should define a CDSS that is implementable within four weeks.

The answer of 2 and 3 triggered the question what a dynamic support entails from their perspective. One of the representatives (2) explained that the concept of dynamic ranges from one extreme to another. In a minimum variant, a dynamic BAIT-based CDSS should be able to run and consulted by the clinical end users working at a specific department. This requires the architecture to define a CDSS that delivers one-way support in which a BAIT-based CDSS does not provide recommendations but does add and update according to real-life choices. In a maximum variant a BAIT-based CDSS constantly provides the clinical end users with recommendations while the clinical end users also add real-life choices. The range of potential variants indicates that different healthcare contexts and clinical end users have diverse preferences regarding the service that the CDSS delivers. This also found in literature (Blank, Graves, Sepucha, & Llewellyn-Thomas, 2006; Eapen, 2021). The architecture should therefore allow Council to build more and less extensive versions of a dynamic BAIT-based CDSS that differ in the completeness of the service delivered to the healthcare client. A fixed architecture is therefore perceived as not useful. Moreover, it is widely acknowledge that the early involvement of clinical end-users in the CDSS development and the full realization of clinical user needs and expectations prior to the development of a CDSS, which is achieved with an adaptable architecture structure, both increase the likelihood of CDSS acceptance (Khairat et al., 2018). Instead, the architecture should allow the guidance of different variants of a dynamic BAIT-based CDSS that match different end user preferences in the different healthcare application contexts of BAIT.

The set of potential wishes and preferences regarding a dynamic BAIT-based CDSS is, however, unknown yet. As specified by Moore and Chang (1980, p. 12), the decision-making contexts as experienced by the end user is “constantly evolving in a decision space whose dimensions include the problem setting, the manager’s preferences, the technology of decision-making and the environment”. The end user’s preferences regarding a support CDSS are dynamic and therefore may change over time. This implies that the specific the set of potential service variants is ambiguous as well. During the design of decision support CDSSs, the discrepancy between the current design and the known as well as the anticipated needs should be minimized (Moore & Chang, 1980). The architecture must therefore enable the effortlessly completion of new modules (variants) that are yet unknown, so it can be extended according to end user needs that become clear in future. Here, effortlessly is defined as without any additional adjustments to the existing content of the architecture. Summarizing the above, the architecture should thus be adaptable to all known and potential future needs in potential healthcare client contexts.

- The architecture should be adaptable to all preferences present in healthcare decision-making contexts.
- The architecture should be adaptable to all potential future preferences in potential healthcare decision-making contexts.

Representative 1 formulated the purpose of the architecture on a more practical level. The focus of this representative was on an architecture that explains the specific steps that are to be taken to both (1) save additional real-life choices made by clinical end users, and (2) update the model with these choices. In essence, the dynamic CDSS should be capable of updating according to new choice information captured by real-life changes to maintain its accuracy.

Representative 3 explained that it is thereby important that the architecture should enable Council to judge if the architecture indeed defines a dynamic BAIT-based CDSS that is capable of updating over time. When asking further about when and how Council wants to judge the updating performance of BAIT, representative 1 and 2 stated that in the beginning it is aimed to analyze different types of models over time in order to understand and gain experience about the updating behaviour of a BAIT-based CDSS or to prepare a client meeting about a (series of) model update(s). To this end, Council would like to have insight in the model diagnostics (Discrete Choice Modeling based metric). Moreover, Council would like to observe all performance metrics - the metrics borrowed from Machine Learning and the Confidence Representation rate - of all model updates of end users over time. Council does not need to be informed about the improvement in model performance as soon as the end user performs an update like the end user, because then Council would be alarmed about updates all the time. This means Council only need to be able to access models as soon as Council wants to.

- The architecture should force the development of a CDSS that gives Council insight into the performance metrics for the choice recommendation generator model updates of all healthcare contexts.

The representative (3) refined this by stating the architecture should not just describe how the new choice information is to be added to the model as one set containing equally important choice observations, but as single choices that are perceived differently in terms of importance by the end user. Here, importance refers to the extent to which an end user wants a BAIT-based CDSS to adjust towards a specific choice. As a consequence, deal with choice sets with choice observations which the end user perceives as differently important for the updates. More precise, the architecture should be sensitive to choices with different levels of importance in the updating procedure that it specifies. The definition of an important choice, however, is to be formulated by the end user and is therefore unknown at the moment of architecture design. This implies that there are no prefixed importance levels or categories of choice observations that can map the choices that are for example commonly considered as more important. The weight of a real-life that influences the effect of that real-life choice on the update is undefined.

The selection of choices that the end user considers as important can be defined as the set of choices of which the end user believes that if the dynamic BAIT-based CDSS gets informed by that set, the CDSS becomes better. What it entails to become better, however, depends on the goals of the end user regarding dynamic BAIT-based CDSS. The set of choices that is perceived relatively important is the set of choices that causes an increase in the level the trust that a BAIT-based CDSS matches the goals of the end user. The architecture should thus enable the end user to specify how much dynamic BAIT-based CDSS is informed by a specific type of choice such that dynamic BAIT-based CDSS realizes the goals of the end user.

- The architecture should force the development of a CDSS that estimates new parameters according to the choice types and weight specification clinical end users selected as soon as the clinical end user deems this model inaccurate or undesired for decision support.

B.1.2 Interview part 2: Functions of the architecture

The second part of the interviews was focused on the specific functions that the architecture should fulfill in order to realize the main purpose of the architecture defined by the representatives of Council.

Automated service support

Since Council delivers similar services to different healthcare clients, tasks part of the dynamic decision support service will be redundant. Representative 1 aims for automating such redundant

activities. Examples are the model estimations for CDSS updating and the calculation of the performance metrics. However, this would impact the relation between Council and end users. Having less contact with the end users might leave end users with the feeling that help is out of reach. Castillo and Kelemen (2013) state that for end users to accept a decision support CDSS and use it effectively, the presence of assistance is vital. Both during the implementation of a CDSS and after the implementation, having a feeling of sufficient support is essential in decreasing anxiety and frustration of end users regarding the CDSS and fosters a positive attitude towards a decision support CDSS (Castillo & Kelemen, 2013). Moreover, Council will become less informed about the user experiences and feedback. Currently Council analyses performance manually and discusses this in a meeting with the end user. During such a meeting, Council learns about the power of the model, its value as perceived by the users, and where improvements can be made.

For Council to serve more end users and scale up in future, automation of redundant tasks will be needed. However, with the importance of end user support in mind, it is argued that the automation should be focused on the tasks that do not involve moments during which Council could help the end user. An example is the calculation and analysis on the performance metrics: the CDSS can automatically generate the values for these metrics such that Council can easily help the end user with interpreting them when needed. Over time, the end user may become more used to the CDSS and may need less help. Then, it can be decided by the end user that some activities are superfluous instead of Council being perceived as negligent. However, the architecture should enhance the end users' feeling that support of Council is always be available. If end users don't understand the results and effects of automated tasks, the architecture should, therefore, enable end users to always request support on the CDSS components and the outcomes of these components.

- The architecture should minimize the number of intervening actions needed from Council that are not requested by the clinical end user.
- The architecture should force the development of a CDSS that enables clinical end users to always request online support on the CDSS components and outcomes these components produce.

Force ease of use

Representative 3 said that the CDSS should be easy to use and easy to understand for clinical end users who have busy working days and have to perform under stressful conditions. The latter notion is covered in appendix B.1.2. Ease of use, however, requires refinement. The ease of use of an AI CDSS determines for the larger extent the user acceptance (Venkatesh, 2000). A CDSS that requires the clinical end user too much time is expected to not meet the end user's goals: "if a CDSS is occupying copious amounts of time that could be used for patient care, then it is not benefiting the user or the patient" (Castillo & Kelemen, 2013, p.322). The ease of use of a CDSS can therefore be considered as inversely related to the complexity of the CDSS (Keil et al., 1995). Rogers and Shoemaker (1971) define the complexity of a CDSS as the degree to which it is perceived as relatively difficult to understand and use. Having the understandability component covered, ease of use here refers to the complexity of the actions that are required to fulfil the end user's temporal goal(s) with the CDSS, like entering the new patient data in the CDSS, updating the model or getting insight in the CDSS performance. The architecture should thus minimize the time and effort that are required of the end user to fulfil the end user's goals with the CDSS.

- The architecture should minimize the time and number of activities that a CDSS requires from clinical end users to fulfil a clinical end user's goals with the CDSS.

Enhance trust

Both representative 2 and 3 explained that the prototype the architecture generates should be implementable in a way that it maximizes the end user's trust in BAIT. This is important for Council to serve the end user on a long term: lack of trust in the AI technology hinders the end user's uptake and usage (Gefen et al., 2003). For AI tools, trust can be a hindering factor but is key in ensuring the acceptance and continuing progress and development (Siau & Wang, 2018). Additionally, trustworthy AI is considered a precondition for a responsible and ethical application of AI tools like BAIT. This is especially essential for the implementation in sensitive areas such as the health sector (Ten Broeke et al., 2021). A clinical end user's level of trust in decision support

CDSSs is affected by the way the CDSS is designed (Alexander, 2006). However, the value of trust is rather abstract and subjective. A requirement including the notion of trust cannot be interpreted in one way and objectively evaluated. This encouraged the translation of the value of trust in an unambiguous requirement.

Formation of initial trust. In the context of BAIT, trust is to be perceived as something that is not only formed at once, but should be developed maintained over time (Siau & Wang, 2018; Ten Broeke et al., 2021). Respectively, a distinction should be made between initial and continuous trust. Initial trust tackles an end user’s perceived uncertainty and risk regarding a new technology (Li, Hess, & Valacich, 2008). The existing a BAIT-based CDSS is already developed in a way that it triggers initial trust (Ten Broeke et al., 2021). To realize the same effect for the dynamic part of the CDSS, the architecture should specify processes that are in line with the formation of initial trust. According to Siau and Wang (2018), this requires the architecture to define updating processes that are transparent, explainable and trailable.

Trialability refers to the opportunity to try something out and see what the effects are. This enables the enhancement of the end user’s understanding of the components and outcomes of the CDSS (Siau & Wang, 2018). The current CDSS already offers trialability. Considering the dynamic part of the CDSS that is subject of design in this research, it would end user’s trust enhance trust if they could try out updates to see the effect. The architecture should thus guarantee that the end user can experiment with an update without ruining the model settings.

The explainability of AI tools refer to the transparency of the components and outcomes as well as the ability to justify them (Siau & Wang, 2018). Transparent means that a CDSS should make its internal change visible, so that end users can track how a CDSS’s recommendation model evolves. However, if the end user cannot comprehend the inner workings of the dynamic CDSS, this transparency is pointless. The more complex AI models become, the less they are capable of self-explanation in an unintuitive way (Hacker et al., 2020). This becomes especially problematic in the healthcare sector where decisions involve patients’ well-being: ethical tensions will occur if clinicians do not understand the CDSS (Alexander, 2006). Updating components and outcomes specified by the architecture should thus be perceived as rather simple instead of complex. This is also in line with the core of Council, which strives towards simple and explainable AI (Ten Broeke et al., 2021). To encourage the end user’s initial trust, the components and outcomes should be understandable by the end user. Understandability, however, is subjective and needs to be standardized in order to measure it and judge the architecture according to it. The architecture should thus guarantee that both the updating processes and the results of these processes can be understood by an end user without any knowledge about Discrete Choice Modeling (DCM), statistics, and Machine Learning (ML). Besides that this enhances trust, an understandable approach is also desired and emphasized by Representatives 2 and 3 such the expansion of the current service matches the key values of Council.

- The architecture should force the development of a CDSS that makes its internal changes transparent for clinical end users.
- The architecture should avoid the development of a CDSS that forces clinical end users to accept a choice recommendation generator model version.
- The architecture should only include components and provide clinical end users with outcomes that a clinical end user without any knowledge about Discrete Choice Modelling, statistics, and Machine Learning can understand.
- The architecture should always be integrable with the service environment of Council.

Formation of initial trust. Initial trust formation is essential for the CDSS adoption, but for the dynamic BAIT-based CDSS that focuses on use over time, it should be nurtured to overcome that the end user’s tendency to use the CDSS will decrease during ongoing use (Gefen et al., 2003; Siau & Wang, 2018). As for the formation of initial trust, continuous trust can be developed with an interpretable and transparent CDSS that can explains its components and outcomes (Siau & Wang, 2018). This is covered in the paragraph above. Second, the extent to which trust is maintained over time depends on the performance of the CDSS – here, of the dynamic part of the CDSS – and requires the CDSS to be always available when needed and be reliable (Siau & Shen, 2003). According to Representative 1, this requires the dynamic CDSS to always be available for updates in context of a dynamic BAIT-based CDSS. Moreover, the updating processes must not hinder other functions of the CDSS when they are running. This will affect the overall trust of the end user in the CDSS as well.

With regard to reliability, Representative 1 argues that the CDSS should be the result of an objective representation of the decision-making context. When end users are able to determine the selection and weight of particular choices to better match their goals, this objective representation is influenced. However, given the subjectively chosen part of the context (captured by the selected and weighted choices), the update should ensure a direct reflection (and thus objective) of the real-world context. To this end, Representative 1 and 2 argue for the inclusion of update processes that are trustworthy and avoid statistical biases.

- The architecture should force the development of a CDSS that only updates the choice recommendation generator model according to contextual changes captured by the experiment choices and real-life choices the CDSS is informed about.
- The architecture should only define components that work statistically correct.

Third, continuous trust will be developed if the CDSS works in collaboration with the end user rather than taking over the end user's tasks (Alexander, 2006; Siau & Shen, 2003). Because collaboration stimulates the perceived control a clinical end user has over the CDSS, it also decreases the perceived threat for the professional autonomy by the clinical end user. Representative 3 argued for an architecture that maintains the feeling of autonomy of the end user. The representative meant that the CDSS should not give a clinical end user the feeling that once implemented, the CDSS takes over the work of the clinical end user. Therefore, the dynamic BAIT-based CDSS should always operate in collaboration with a clinical end user. The representative expected that although the tasks of a dynamic BAIT-based CDSS do not directly need or replace the knowledge and skill of a clinical end user, clinical end users would still like to have the ability to control it such they know what is going on with regard to the CDSS. This is tested during the interviews with the clinical end users. A similar feeling of ownership can be threatened if Council has too much influence. For example, Council manually assesses the model performance and discusses this with the end user. This is out of control of the end user. The enhancement of trust in the CDSS will benefit from a minimization of the intervening actions performed by Council that are beyond the control of the clinical end user, and thus are not requested by the end user. Besides the perceived autonomy, this also ensures the minimization does not touch support requests of end users.

Finally, data security and data privacy play a key role in nurturing the trust in a new AI CDSS (Kusumasondjaja et al., 2012; Siau & Shen, 2003). The architecture forms an expansion of an existing CDSS for which security and privacy considerations are well organized. The architecture should therefore include and build upon the saving processes and applications that are used by the current CDSS. The procedure for the current CDSS and associated services states that information should not be retrievable to individuals. Moreover, the data sharing with Council is organized using the Information Processing Agreement they already have in place.

- The architecture should force the development of a CDSS that clinical end users can always use for a choice recommendation request and measurement of a real-life choice.
- The architecture should always contain processes that work in partnership with clinical end users.
- The architecture should ensure choices are not retrievable to an individual clinical end user and Council only has access to decision-making behaviour for which end users provided permission.
- The architecture should minimize the number of intervening actions needed from Council that are not requested by a clinical end user.
- The architecture should avoid the development of a CDSS that forces clinical end users to accept a choice recommendation generator model version.

B.1.3 Interview part 3: Characteristics of the architecture

The final part of the interviews was focused on the characteristics that the architecture should have in order to realize the purpose of the architecture. In order to elicit this information. The main characteristics identified during the interviews is that the architecture should be complete and compatible. Both characteristics is further explained below.

Completeness of the architecture

Representative 1 mentioned that the architecture should represent all aspects that are part of the CDSS expansion, as well as its connections to the current CDSS. The main reason provided is that developers over time need to understand the architecture, also without guidance of persons involved in the architecture design process. To this end, the architecture should assume that the developer will overlook every required aspect during the CDSS development process that is not defined by the architecture. Moreover, the architecture should be complemented with materials that explain and argue for all aspects that are defined by the architecture.

- The architecture should inform how the architecture is used for development in specific healthcare decision-making contexts.
- The architecture should inform on all processes that are needed to achieve the goal of the CDSS.
- The architecture should inform on all data objects associated with the architecture.
- The architecture should inform on all dependencies between architecture components.

Compatibility of the architecture

Representative 2 emphasized that the architecture forms an expansion of the current CDSS that is already used to provide a particular service. This service is grounded in a particular organizational and technological environment. To let the architecture be a useful starting point for CDSS development for Council, all organizational and software processes and components it defines must fit into the current environment. Otherwise, it will not be possible to connect the expansion that the architecture defines with the existing CDSS service. The technological environment is an online coding platform (WEM). This platform is a software that deals with the data storage and does not require any particular hardware tools rather than a computer to access the platform. Organizational wise, Council aims for a quick CDSS development and has already privacy procedures in place (covered in appendix B.1.1 and appendix B.1.2).

- The architecture should always be integrable with the service environment of Council.
- The architecture should define a CDSS that is implementable within four weeks.
- The architecture should avoid the retrievable of choice information to an individual clinical end user and undesired access to decision-making behaviour in a particular context by Council.

B.2 Interviews clinical end users

B.2.1 Interview part 1: The design of the updating

The end users were first asked regarding their perspective on the importance of updating a BAIT-based CDSS with new real-life choices. The reason was to check whether the clinical end users gain trust from choices made in real life, next to choices made in a controlled, situated experiment. All clinical end users appeared to be convinced that a BAIT-based CDSS should update according to real-life choices for the following reasons:

1. The healthcare sector changes fast, and a decision support CDSS should be flexible towards this change. For example, choices regarding the ICU uptake of COVID patients were made with different knowledge and based on other criteria a year ago than they are now.
2. Clinical end users trust in a learning CDSS rather than a CDSS that is based on a set of parameters that is perceived as always valid. This better mimic the clinical end user who also learns and adjusts its strategy over time.
3. Every decision and the success of its outcome inform and influences future ones. Such feedback loops are essential and should be mimicked by a dynamic BAIT.
4. The choice experiment is only conducted with a selected set of clinical end users.

The potential “noise” that may result from decision-making in a stressful context (see section 2.2.2), did not make clinical end users who were interviewed anxious of adding real-life choices. They explained that this noise is part of the real-world, and therefore part of the decisions that clinical end users take, and should thus be experienced by a BAIT-based CDSS as well.

clinical end user 3 added that the CDSS should not just be able to attune according to new real-life choices, but that it would be valuable if the CDSS is also flexible enough to adjust to radical changes. It happens that a new treatment is developed that completely changes the decision-making. The illustration the clinical end users provides was the introduction of the COVID vaccine that influenced the decision-making on the ICU uptake. As such, the value of the CDSS would increase if the CDSS is not only able to slightly attune its parameters over time by reflecting gradual contextual changes, but is also modifiable in a way that radical changes can be reflected as well.

Choice inclusion and weighting

clinical end user 2 prefers to include every real-life choice entered in and associate every real-life choice with an equal weight. The clinical end user further specified that it is good that the real-life choice will form the bigger part of the database and therefore will become relatively important over time. From this notion it can be derived that the clinical end user wants a BAIT-based CDSS to rely more on the real-life choices than on the experiment choices, since the real-life choices better represent the real-world and the present status-quo. The clinical end user explained that, therefore, it would not be needed to modify the weight of choices. clinical end user 3 gives a similar answer and adds that the situation in which individuals are determining the weights of single choices is not desirable, because then the BAIT-based CDSS will follow a subjective preference. The subjective power in the learning of a BAIT-based CDSS should, instead, be minimized. The following example was given: if I have a bad start of the day, had a fight with my wife, it was raining, and had a bad coffee this will influence my opinion regarding the choice I made, and whether it is of added value to the update of the model. According to clinical end user 3, this subjectivity should be excluded from the model, because such subjectivity steers the model in a way that is not fair to the patient.

clinical end user 1 agrees with 2, but specifies that it might be desirable that not all real-life choices are included. The clinical end user explains that it is goal-dependent whether a choice should be included or not. If an end user wants the model to perfectly represent the real-world, for example for end users to reflect on their own decision-making, it is desired to include all real-life choices and do such they are equally relevant. The resulting model may be sub optimal because it is based on imperfect real-life choices, for example real-life choices that were the result of a stressful decision-making context in the middle of the night but does approach the real-world as closely as possible. On the other side, if an end user could desire a model that only includes real-life choices that meet specific criteria, for example that it results from decision-making of an experienced senior clinical end user in between nine in the morning and five in the afternoon. It could also be preferred that the group of recently added, and therefore, most accurate real-life choices, have a larger weight. Another option would be to let the influence of experiment choices and early real-life choices lower over time. According to clinical end user 3, the resulting model is then considered more optimal and qualitative than objective, since it is based on a specific set of “safe” real-life choices that meet specific optimal criteria.

The view points and reasoning with regard to the inclusion and weighting of different choices in the update varied among the clinical end users who were interviewed. Moreover, because the interview findings represent only three clinical end users, more viewpoints may exist in a larger pool of clinical end users. Because preferences may also vary over time, the architecture should specify all possible inclusion and weighting options. To enable the different weighting options, the experiment choices and real-life choices should be stored separately. Besides the weighting of real-life choices relative to other real-life and experiment, a final notion was made regarding the set of experiment choices by clinical end user 1. Although the influence of this group of choices will decrease over time, the clinical end user shared that it would be good if the model could be informed with new choices that were not (partly) informed by the recommendation of the CDSS as is the case for real-life choices. Choices that for which it is sure that they are not informed by the CDSS, are choices made in a controlled experiment. However, because these choices are made in a controlled environment with the use of a specific survey design they should not be combined with choices made in another controlled environment. As such, the CDSS should accept the replacement of new experiment choices rather than the addition of new experiment choices. If the end user

would like to enlarge the influence of experiment choices this should thus be done by giving these choice types a larger weight instead of adding up new choices collected with different experiments.

- The architecture should force the development of a CDSS that estimates a new choice recommendation generator model according to the choice types and weight specification clinical end users selected as soon as clinical end users deem this model inaccurate or undesired for decision support.
- The architecture should force the development of a CDSS that distinguishes experiment choices and real-life choices.
- The architecture should avoid the development of a CDSS that allows clinical end users to directly determine the importance of a single real-life choice in the model estimation.
- The architecture should force the development of a CDSS that allows replacing the experiment choices with experiment choices from a new choice experiment.
- The architecture should force the development of a CDSS that stores a choice with all features according to which that choice is specified.

Value of Objective Clinical Outcomes for update

The interview continued with the perceived importance of clinical objective outcomes (OCO's). An OCO represents the consequences on the patient's well-being of a choice. clinical end user 3 immediately states that OCO's are really important, because they function as an evaluation of the decision reasoning of the CDSS. The following example was given: if a clinical end user chooses to operate nine out of ten patients, but for eight out of the nine patients the operation did not make the patient well-being better or even made it worse, than this information must be provided to the CDSS as feedback. clinical end user 1 adds clinical end users themselves also learn from OCO's. When the effect of a specific choice is undesired, this will inform the choices to be made in future. According to clinical end user 3, currently there is a vague procedure for collecting OCO-like information. When a BAIT-based CDSS allows to adopt such information, this will enhance both the updating of a BAIT-based CDSS and ourselves. However, clinical end users find it yet hard to quantify how important. Since the clinical end users consider an OCO as ground truth, it should at least always have a larger effect than a real-life choice has on the model update.

Dealing with incomplete choices for update

The experiment choices have a prefixed format. End users that enter new real-life choices have to do this in a similar format, meaning an end user has to specify all the criteria for a patient. In the real-world, however, some patient data may be unknown, unclear or delayed. Moreover, the complexity of the choice tasks, referred to as information load as source of cognitive burden, may be high. clinical end users may ignore specific criterion as a coping strategy in order to deal with the perceived complexity (Hensher et al., 2005). This means the criterion was not relevant for the specific clinical end user given the values for the other criteria in the specific choice task. It would then result in more realistic reflection of the context if the end user could tell a BAIT-based CDSS about the irrelevance of the criterion in the specific choice task. The CDSS already allows end users to inform the CDSS that some data is missing. In the calculation of the recommendation, the CDSS then calculates the recommendation so that the utility contribution to the recommendation of the unknown criteria is zero. A dynamic BAIT-based CDSS does not only have to calculate a recommendation based on data with unknown values, but also has to estimate parameters for a new update. This links to a recognized issue in DCM literature: in the estimation of choice models based on real-life choices, information on the choice set may be missing or incomplete or data on some alternatives may be lacking (M. Ben-Akiva et al., 1997).

The clinical end users did not provide a clear answer on how the CDSS should deal with missing data. clinical end user 1 and 3 showed their trust in the designer to come up with a statistically valid solution. clinical end user 2 specified that it should at least be noted that patient data cannot be interchanged for example by filling missing values with averages calculated over other patients. One of the approaches to obtain information on missing attribute values is to find solutions by which they are imputed with the average of observed values (Washington et al., 2014). However, because the medical data is very specific for a particular patient, it cannot and should not be replaced with other information that stems from other patients. Moreover, since the flow of incoming real-life choices is marginal the data over which an average value would then be

calculated is too limited (Washington et al., 2014). The average will be too sensitive to extreme values of other patients. As an alternative to imputing missing values with the average, Steinberg and Scott Cardell (1992) show that it is possible to estimate choice models by pooling the accessible real-life choices with data that is publicly available. However, again because a BAIT-based CDSS is applied in healthcare contexts where the data is sensitive but also patient specific this is not valid. This means there has to be dealt with missing values that cannot be replaced with other data sources.

- The architecture should force the development of a CDSS that does not interchange patient-specific data to deal with incomplete real-life choices.

B.2.2 Interview part 2: Frequency and activation of updating

The second part of the interview was focused on how the updating should be controlled. No particular conditions on the availability of an update were found. With regard to the level of updating automation, clinical end user 2 states that the updates should not be activated automatically. The idea that the model slightly changes outside the clinical end user's control gives a feeling of anxiousness. The other two clinical end users (1 and 3) see advantages in an automatic updating component. However, they always want to know when and understand how the model changes. Moreover, they want to be able to reset the model that is active if they deem a new model update as undesired. Therefore, clinical end users should always be in the position to select the choice recommendation generator model that the CDSS operates. The need of control regarding the updating, either by activating it or fully understanding it, shows that clinical end users do not directly believe that a updated model is always better and want to judge this themselves. This confirms the importance of perceived autonomy as expected by Councyl (see appendix B.1.2). Clinical end user 1 adds that it would be valuable to first execute updates manually and gain experience with the dynamic CDSS and the effects, to later automate the update activation if trust in the CDSS has been developed. Because the viewpoints already vary within the group of three clinical end users, and preferences may change over time, the architecture should define different options for the level of updating automation. Also literature indicates that full automation is not similarly desired in every healthcare context (Aron et al., 2011; Eapen, 2021; Khairat et al., 2018; Pirnejad et al., 2019; Wang et al., 2021).

However, when fully leaving the updating of a BAIT-based CDSS to clinical end users, there is a risk that updates are forgotten or that clinical end users are unsure whether an update can be done already (Smith et al., 2009). The architecture should thus ensure that the activation is somehow encouraged, while giving the clinical end users the feeling they have the power in the updating procedure. To this end, the architecture should always define a trigger to encourage the clinical end user to activate a new model update as soon as the performance has degraded below an accepted level.

For both the automated activation of a model update and the trigger towards the end user, insight in the level of acceptance is needed. The clinical end users gave slightly different answers with regard to the accepted level of performance. The value for the level of acceptance should therefore be modifiable per context.

- The architecture should never force a fully automated CDSS.
- The architecture should avoid the implementation of a CDSS that forces a clinical end user to accept a choice recommendation generator model version.
- The architecture should force the development of a CDSS that estimates a new choice recommendation generator model according to the choice types and weight specification clinical end users selected as soon as clinical end users deem this model inaccurate or undesired for decision support.

B.2.3 Deviating choices

With respect to what a BAIT-based CDSS should inform the end users about, clinical end user 2 and 3 both showed interest in notifications regarding interesting cases. When asking further what interesting in this context means, the clinical end users explained that they would like to know for which choices a BAIT-based CDSS was confident, for example a clear majority agrees in favour or against the operation, but the clinical end user made another decision. It is outstanding if a

BAIT-based CDSS represents a high internal agreement regarding a specific decision case among the pool of experts, for example a recommendation of 95% in favour of operating, while the clinical end user entering the real-life choice makes a different decision and for example decides to not operate). According to the clinical end users, this points to the cases where either the clinical end user or the BAIT-based CDSS differs from a specific line of reasoning in the decision-making. clinical end user 1 was also interested in this type of information but wants it to be stored. This would allow to compare the cases for which the BAIT-based CDSS was sure but did not agree with the clinical end user and investigate what the causes may be.

- The architecture should force the development of a CDSS that compares each choice recommendation with the majority threshold and the clinical end user's choice as soon as an end user enters a real-life choice into the CDSS.
- The architecture should force the development of a CDSS that copies real-life choices exceeding the majority threshold but deviate from a clinical end user's choice to a separate database.

Desired information provision

The information clinical end users want to receive from the CDSS is summarized in four main subjects: information on the decision-making behavior, the internal model changes, the updating activity, and the model performance. The main reason which holds for all three subjects it that the clinical end users emphasized that, in order for them to trust the CDSS and its updating process, they want it to be as transparent as possible. With transparent they meant that they trust the model builder with the technical details, like dealing with missing values, but find it important to know when a CDSS executes an update and what the results of an update are. Below the requirements related to the three subjects are respectively further explained.

1. Identified requirements with respect to the decision-making behaviour of the clinical end users is addressed in appendix B.2.5.
2. According to clinical end user 1, the transparency of the internal model changes due to updates is key. All clinical end users want to have insight in the model that is used and how this model changed after an update. This allows clinical end users to judge whether the model that results from the update is desired. For example, it could be that age has become a more important factor in the decision-making of the decisions while this is actually not desired. However, something that stood out was that clinical end user 2 argued that the model information should not be used as explanation for choice recommendation when a clinical end user requests a choice recommendation. When requesting choice recommendation, the clinical end user is occupied and does not have the time and focus to interpret the explanation. Moreover, the clinical end user should not be distracted by what the pool of clinical end users find important. It should rather be accessible as soon as it suits the end user and the end user wants to interpret the model with a specific purpose. Moreover, clinical end user 1 mentioned that it is desired to compare model updates rather than just interpret the model that currently is used for the choice recommendation generation. clinical end users should therefore always be able to request insight in all model updates.
3. Information on the updating activity refers to the confirmation of the activation and completion of an update.
4. Identified requirements with respect to the information on the model performance is addressed in appendix B.2.4.

As result, the requirements identified are:

- The architecture should force the development of a CDSS that allows clinical end users to request the relative importance of the choice attributes of all model updates.
- The architecture should force the development of a CDSS that confirms the completion of a choice recommendation generator model update.
- The architecture should force the development of a CDSS that presents an alert to clinical end users when the level of acceptance has been reached and the choice recommendation generator model is not updated yet.

B.2.4 Information provision on the model performance

Information on the model performance is twofold. First, the clinical end users find it important to be informed as soon as the performance of the model based on which choice recommendation is generated has degraded below a level of acceptance. This was also found to be important to ensure clinical end users are triggered to activate updates over time (see appendix B.2.2). Because clinical end users are occupied during working hours, alerts that need action should remain visual until the associated action has been successfully performed (Khalifa, 2014).

Second, next to judging the change in the relative importance of choice attributes in a model update, all clinical end users emphasize they want to know to what extent the performance of the CDSS has improved or decreased. Therefore, they need to be informed on the performance of the new model for choice tasks in the new, changed context. The clinical end users all stressed that being informed the accuracy (Correspondence rate) of the model and do not want too much information. All clinical end users also emphasized that the metrics that define the model performance should be calculated based on recent data points. The healthcare decision-making contexts in which the end users operate are highly dynamic, meaning that the conditions under which they make their decisions change fast. For example, if a model that is update in January 2021 has a high accuracy when being tested on real-life choices that were entered into the CDSS in March 2020 this does not indicate that the model update leaves the end user with an accurate CDSS. clinical end user 3 stated that the metrics have no value if they are based on validation data that includes cases of March 2020. Therefore, the data set based on which the CDSS calculates the metrics should only take into account real-life choices that were recently added to the database of the CDSS to ensure the outcomes are meaningful to the end user.

Since recent is a rather subjective concept, this requirement needs to be further specified. As the real-life choices that were added latest to the CDSS are the most recent ones, the architecture should ground the metrics on a minimal number of the real-life choices that were entered latest. Because some decision-tasks are rare, filling the test set with the minimal number of real-life choices will lead to a set with real-life choices that cover a relatively long period. For contexts that are relatively static, this is not a problem. However, it might lower the value of the metrics for end users in contexts that change fast. Because no other data is available, no alternative to create the test set exists. The end user should thereby at least be able to judge to what extent the metrics calculated are of value and to what extent the end user judges the model performance based on these metrics. Therefore, the architecture should make the date transparent on which the real-life choices that were used for the calculation of the metrics were entered in the CDSS.

When further discussing the content of the data set based on which the performance of the model is determined, all clinical end users agreed that experiment choices should also not be used for the model validation. The clinical end users have less trust in the metrics if they are grounded on data collected in a controlled experiment setting. Instead, they are interested in the performance of the model on real-life cases.

- The architecture should force the development of a CDSS that allows clinical end users to request the relative importance of the choice attributes of all choice recommendation generator model updates.
- The architecture should force the development of a CDSS that makes the date at which a clinical end user entered a choice used for the model validation transparent.
- The architecture should force the development of a CDSS that assesses the performance of a newly estimated choice recommendation generator model based on a unique set of recent real-life choices.
- The architecture should force the development of a CDSS that enables clinical end users to request the performance metrics for all choice recommendation generator model updates.

B.2.5 Additional decision-making behaviour insights

All three clinical end users appeared to be interested in the advanced decision-making trends. According to the clinical end users, this allows for a so-called ‘mutual learning’. clinical end user 3 explained that this enables clinical end users to learn from each other: in the medical world experienced senior clinical end users tend to be confident about their reasoning and could benefit from opening up to juniors who are educated according to the latest technologies. A dynamic BAIT-based CDSS could shed light upon the differences between the decision-making of seniors and juniors. clinical end user 1 stated that every insight around the decision-making of clinical

end users would be of value. More specific, the patterns all clinical end users were interested could be listed as follows:

1. The differences between choices that were made by seniors versus juniors
2. The differences between choices that were made by different expertise
3. The differences between choices that were made by different disciplines
4. The differences between choices about which the CDSS was confident
5. The differences between choices about which the clinical clinical end user was confident
6. The differences between choices made in the choice experiment setting (experiment choices) and the real-world (real-life choices)
7. The differences between choices about which the expert was certain versus uncertain
8. The pattern according to which the relative importance of a decision criteria changes over time

Such insights require meta-data associated with the real-life choice. This gives rise to a trade-off between the potential extensiveness of the insights in the decision-making patterns and the amount of data that is to be entered by the user. Although all clinical end users acknowledged to have an aversion to entering large amounts of data, they would feel encouraged to do so when the data requested is really needed for the purpose of the advanced insights. The clinical end users should thus have the feeling that the extra information they enter is of value later on. Moreover, the extra effort asked from clinical end users should be minimized while still being able to get the advanced insights. Therefore, the CDSS should not request the same data each time that is the same for each real-life choice. For example, the function of the clinical end user. Instead, this should be associated to the profile of the end user that enters the real-life choice.

- The architecture should force the development of a CDSS that allows clinical end users to request the relative importance of the choice attributes for a by the clinical end user selected subgroups.
- The architecture should force the development of a CDSS that only asks a clinical end user to enter choice-specific data when entering a real-life choice from which the clinical end user will benefit later in time.
- The architecture should force the development of a CDSS that stores a choice with all features assigned to the choice when a clinical end user entered the choice into the CDSS.

B.2.6 Majority threshold

For clinical end user 1, a BAIT-based CDSS should predict that a majority of 80% votes in favour of the operation for him to consider it as a serious recommendation in favour of the treatment. clinical end user 3, however, only needs 50%. According to clinical end user 3, decision-making works like a democracy. If the majority is convinced that proceeding with a treatment, the final decision will be to operate. clinical end user 2 did not have a clear opinion yet but was not directly convinced by the idea of being informed by just a small majority. Clearly, the opinions regarding the threshold are diverse. Moreover, the opinion of an end user regarding the threshold may change over time. Instead of formulating a fixed threshold, a variable threshold that is chosen by the end user and can be adjusted over time seems to be a better fit. This requires a BAIT-based CDSS to be sensitive to different thresholds when informing the end user on a deviation. Because the CDSS needs to accept values to be determined by the end user, an environment where end users enter these values is required.

- The architecture should force the development of a CDSS that operates a modifiable majority threshold and level of acceptance.

Appendix C

Archimate legend and relationship description

This appendix presents the specification of the Architecture Description Language ArchiMate. The following link directs to a page that provides the specification of all ArchiMate elements, including all components and all relationships between the components: <https://pubs.opengroup.org/architecture/archimate3-doc/>. Table C.1 gives an overview of the most common relationships between architecture components and the definitions of all relationships. Table C.2 presents additional specifications and the definitions of the shortcut words used in the architecture for Council.

Table C.1: *Definition of relationships specified ArchiMate.*

Relationship	Definition
Composition	The composition relationship represents that an element consists of one or more other concepts.
Aggregation	The aggregation relationship represents that an element combines one or more other concepts.
Assignment	The assignment relationship represents the allocation of responsibility, performance of behavior, storage, or execution.
Specialization	The specialization relationship represents that an element is a particular kind of another element.
Realization	The realization relationship represents that an entity plays a critical role in the creation, achievement, sustenance, or operation of a more abstract entity.
Used by	The used by relationship represents that an element provides its functionality to another element (also referred to as the serving relationship).
Access	The access relationship represents the ability of behaviour and active structure elements to observe or act upon passive structure elements.
Association	An association relationship represents an unspecified relationship, or one that is not represented by another ArchiMate relationship.
Triggering	The triggering relationship represents a temporal or causal relationship between elements.
Flow	The flow relationship represents transfer from one element to another.
Junction	A junction is used to connect relationships of the same type.

Table C.2: *Additional specifications and shortcut words.*

Reference	Definition
"(Copy)"	Explicates that the original component is located elsewhere in the architecture, but for visibility purposes has been copied and replaced.
Black dot	Marks processes that should run in parallel.
"DB"	Database.
"Recommend."	A CDSS's choice recommendation.
"User ID"	Identified of logged in user.
"temp"	Temporary database.

Appendix D

Architecture current situation

This section presents the architecture of a BAIT-based CDSS that is currently used to serve clinical end users by Councyl (see fig. D.1).

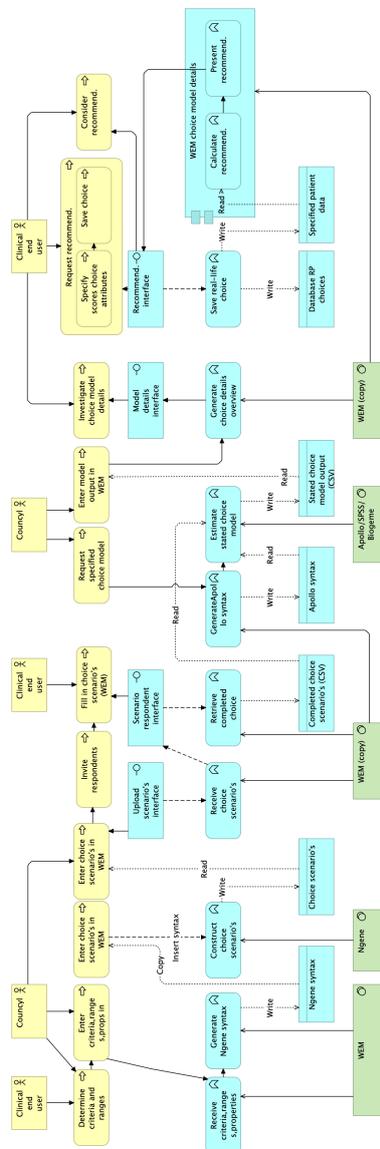


Figure D.1: The architecture of a BAIT-based CDSS used to serve clinical end users by Councyl.

Appendix E

Architecture solution for Council: full overview

This section presents the complete architecture of a dynamic BAIT-based CDSS that is designed for Council in full detail (see fig. E.1). The Information specification Extensions are presented on the next page (see fig. E.2).

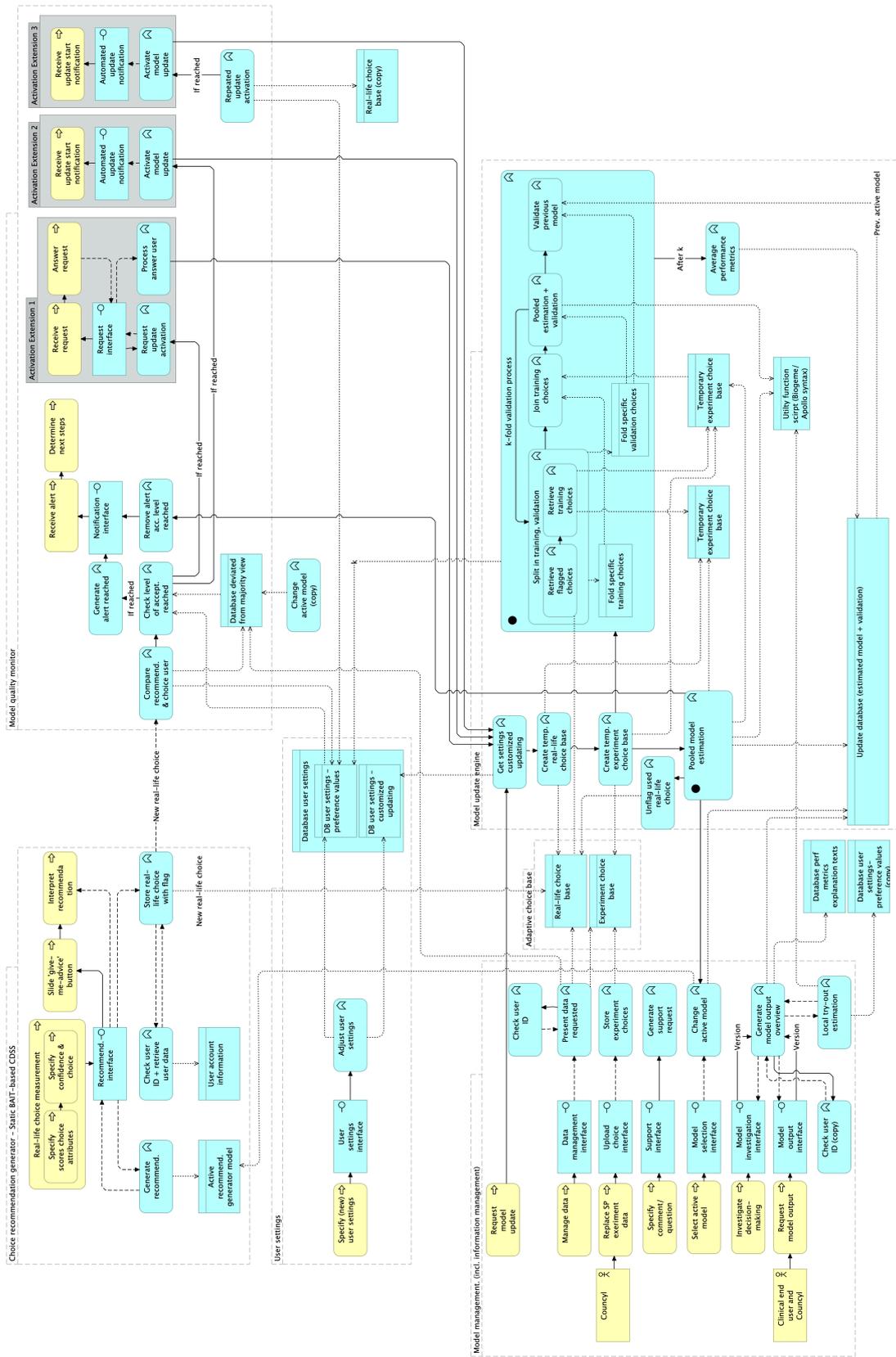


Figure E.1: The architecture of a dynamic BAIT-based CDSS for Cuncyl.

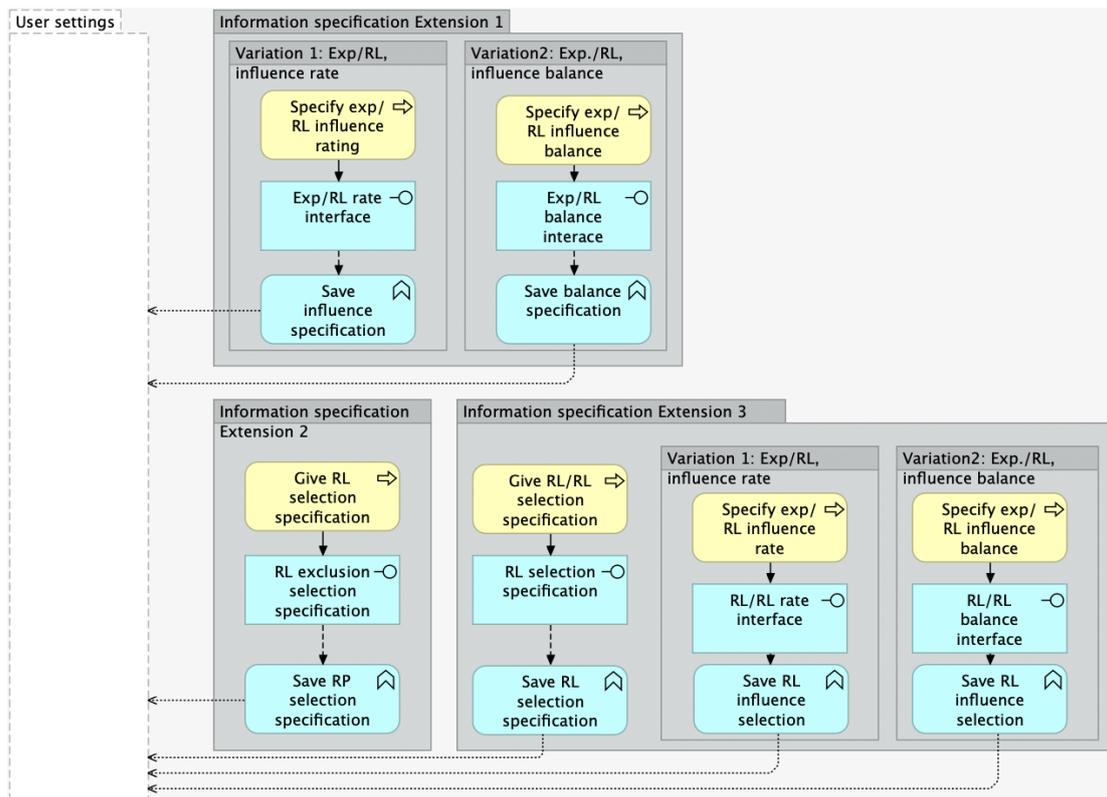


Figure E.2: The Information specification Extensions that are part of the architecture of a dynamic BAIT-based CDSS for Council.

Appendix F

Architecture solution for Council: high-level overview

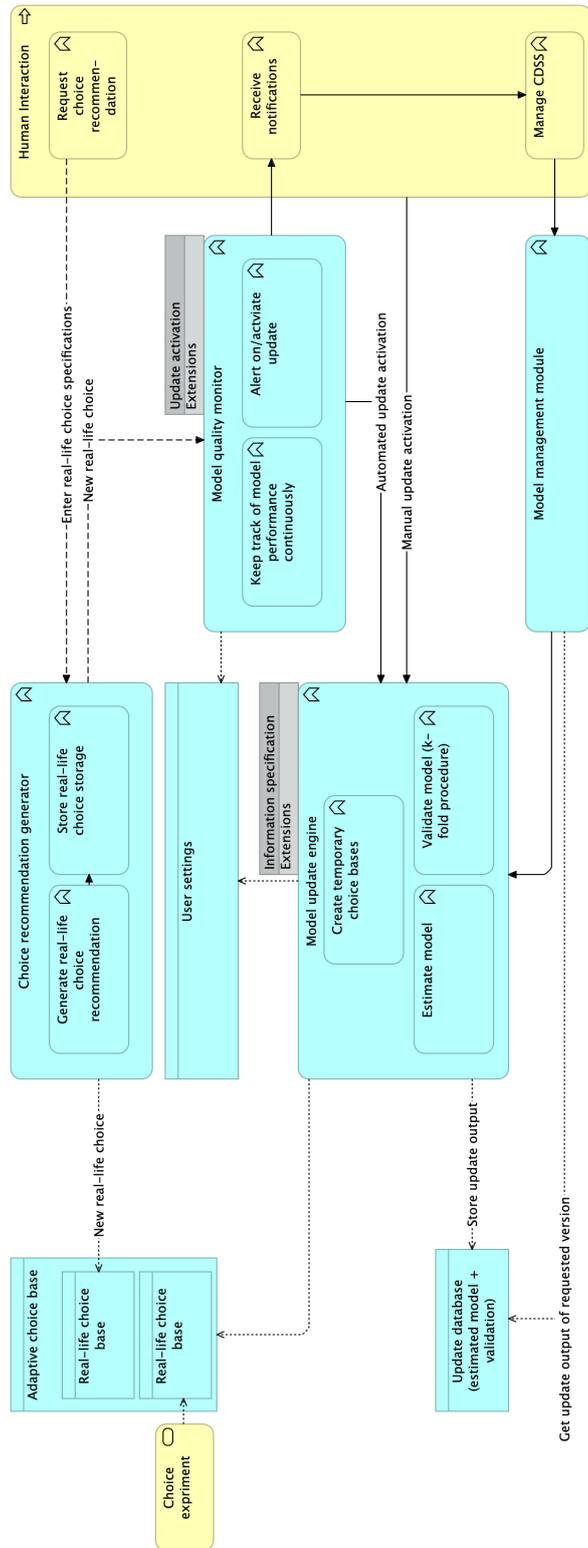


Figure F.1: High overview of the dynamic BAIT-based CDSS architecture for Council1 with six components.

Appendix G

Architecture proof-of-technology: scripts and outcomes

G.1 Scripts proof-of-technology

This appendix presents the scripts and outcomes that were used during the evaluation of the proof-of-technology. The goal of the architecture is to describe the components that are to be performed such a BAIT-based CDSS can be updated according to new choices. To customize these components according the preferences of clinical end users, the architecture defines a set of Information specification Extensions that modify the way in which the update is executed, each satisfying different end user preferences. The goal of the proof-of-technology is to assess whether Council is convinced that the technical updating components specified in the architecture generate the intended outcomes, and whether the outcomes are correct. To this end, the model update engine and the three Information specification Extensions were implemented. An explanation of these components is given in table G.1.

Table G.1: *Overview table of the implemented and tested architecture components*

Component	Explanation of component functionality
Component 1: The model updating engine	The estimation and validation of a recommendation generator model with experiment choice and new real-life choices. The process part of this component are needed for every clinical end user. The estimation should result in a set of parameters for all choice attributes and an error term. The validation should indicate the performance on new real-life choices in terms of the Correspondence rate and the Recommendation-choice Agreement table (also referred to as the Confusion Matrix).
Component 2: Information specification Extension of updating engine 1	The estimation and validation of a DCM in which the importance of experiment choices and real-life choices are varied with both a importance rate and balance (see section 4.3.1). For instance, when a clinical end user values experiment choices two times more important than real-life choices.
Component 3: Information specification Extension of updating engine 2	The estimation and validation of a DCM in which a particular set of real-life choices are excluded. For instance, when a clinical end user wants the CDSS to only incorporate choices made by senior clinical end users. The choice base that is used for the estimation should only include the selected choices.
Component 4: Information specification Extension of updating engine 3	The estimation and validation of a DCM in which the importance of a particular set of real-life choices is differently weighted compared to the remaining real-life choices by using both an importance rate and balance specification. For instance, when a clinical end user considers choices made by senior clinical end users as more important than choices that were made by junior clinical end users.

The implementation was done with the use of a python package called Biogeme, which relies on the package Python data analysis library Pandas. It is an open-source package designed for the maximum likelihood estimation of parametric models in general with an emphasis on discrete choice models as needed for the implementation (Bierlaire & Fetiariison, 2009). It contains all

the required methods to estimate choice models. The scripts are built for test data. The test data set contains experiment choices and real-life choices collected in a healthcare context. The represented decision task is the choice in favour or against ICU uptake of COVID patients. The answer type was twofold: no ICU uptake and ICU uptake. The data set represents 502 choices: 425 experiment choices and 77 real-life choices. The real-life choices were made in the period from March 2020 to October 2020. This choice data was enriched with dummy data. This dummy data contains additional CDSS features to the choice samples that are not collected yet by the current static BAIT-based CDSS.

The scripts used for the proof-of-technology are presented in fig. G.1 to fig. G.7. A brief explanation of the code is provided within the scripts as well as in the captions of the figures. The following abbreviations are used in the scripts:

- SP = Stated preference = Experiment choice
- RP = Revealed preference = Real-life choice

```
In [8]: import pandas as pd
import math
import biogeme.database as db
import biogeme.biogeme as bio
import biogeme.models as models
import biogeme.version as ver
from biogeme.expressions import Beta

import pandas as pd
import numpy as np
import biogeme.database as db
import biogeme.biogeme as bio
import biogeme.models as models
import biogeme.optimization as opt
import biogeme.results as res
from biogeme.expressions import Beta, DefineVariable
#from sklearn.model_selection import train_test_split

from ipywidgets import interact, interactive, fixed, interact_manual
import ipywidgets as widgets
```

Figure G.1: The python packages needed for the proof-of-technology. Required for implementation of all components in table G.1.

Insert choice data CSV

```
In [9]: csv_file_choices = "Choices.csv" #specify name
choice_data_pandas = pd.read_csv(csv_file_choices, sep=',')

choice_data_pandas.drop(["SET",
                        'ID',
                        'a_desc_Alt1',
                        'b_desc_Alt1',
                        'c_desc_Alt1',
                        'd_desc_Alt1',
                        'e_desc_Alt1',
                        'original_e_Alt1',
                        'f_desc_Alt1',
                        'original_f_Alt1',
                        'g_desc_Alt1',
                        'original_g_Alt1',
                        'h_desc_Alt1',
                        'original_h_Alt1',
                        'i_desc_Alt1',
                        'original_i_Alt1',
                        'j_desc_Alt1',
                        'original_j_Alt1',
                        'k_desc_Alt1',
                        'l_desc_Alt1',
                        'k_desc_Alt1',
                        'm_desc_Alt1',
                        'n_desc_Alt1'], axis = 1, inplace=True)

#Replace , with a . for python to transform into numeric
choice_data_pandas["c_Alt1"] = choice_data_pandas["c_Alt1"].str.replace(",",".")
choice_data_pandas["e_Alt1"] = choice_data_pandas["e_Alt1"].str.replace(",",".")
choice_data_pandas["f_Alt1"] = choice_data_pandas["f_Alt1"].str.replace(",",".")
choice_data_pandas["g_Alt1"] = choice_data_pandas["g_Alt1"].str.replace(",",".")
choice_data_pandas["h_Alt1"] = choice_data_pandas["h_Alt1"].str.replace(",",".")
choice_data_pandas["i_Alt1"] = choice_data_pandas["i_Alt1"].str.replace(",",".")
choice_data_pandas["j_Alt1"] = choice_data_pandas["j_Alt1"].str.replace(",",".")
choice_data_pandas["k_Alt1"] = choice_data_pandas["k_Alt1"].str.replace(",",".")
choice_data_pandas["m_Alt1"] = choice_data_pandas["m_Alt1"].str.replace(",",".")
choice_data_pandas["n_Alt1"] = choice_data_pandas["n_Alt1"].str.replace(",",".")

#Transform object types into numeric
choice_data_pandas["c_Alt1"] = pd.to_numeric(choice_data_pandas["c_Alt1"])
choice_data_pandas["e_Alt1"] = pd.to_numeric(choice_data_pandas["e_Alt1"])
choice_data_pandas["f_Alt1"] = pd.to_numeric(choice_data_pandas["f_Alt1"])
choice_data_pandas["g_Alt1"] = pd.to_numeric(choice_data_pandas["g_Alt1"])
choice_data_pandas["h_Alt1"] = pd.to_numeric(choice_data_pandas["h_Alt1"])
choice_data_pandas["i_Alt1"] = pd.to_numeric(choice_data_pandas["i_Alt1"])
choice_data_pandas["j_Alt1"] = pd.to_numeric(choice_data_pandas["j_Alt1"])
choice_data_pandas["k_Alt1"] = pd.to_numeric(choice_data_pandas["k_Alt1"])
choice_data_pandas["m_Alt1"] = pd.to_numeric(choice_data_pandas["m_Alt1"])
choice_data_pandas["n_Alt1"] = pd.to_numeric(choice_data_pandas["n_Alt1"])

choice_data_pandas.rename(columns={"CHOICE": "CHOICE", " Scenario IntID": "patientID"}, inplace=True)

#Get SP and RP seperately
SP = choice_data_pandas[:425]
RP = choice_data_pandas[425:]

#Give novelty flag to new RP's
column_flag = {'Novelty_flag':[1]}
novelty_flag_tracker = pd.DataFrame(column_flag, index=RP.index)
```

Figure G.2: The code to retrieve and prepare the choice data in the adaptive choice base for an update. Required for implementation of all components, but represents component 1 in table G.1.

Default model update engine (from here, repeat k times)

```
In [12]: #Construct empty beta and accuracy list to summarize after k repetitions
beta_list = []
accuracy_list = []

for i in range(k):
    #Prepare estimation and validation set
    #Make validation data by getting in a manipulated way: get observations of all patients
    choice_data_pandas_test = pd.DataFrame()
    #while (len(choice_data_pandas_test) + len(multiplied_RP_patientID.unique().tolist()))
    #optional: as long as not bigger than 1/3. Maar hier niet meer unieke observaties
    for i in RP.patientID.unique().tolist():
        choice_data_pandas_test = choice_data_pandas_test.append(RP[RP["patientID"] == i])
    #dezelfde patienten hebben hetzelfde nummer dus hier wordt van elke patient een random

    #Make estimation data based on either SP or RP and RP observations
    if SP_estimation == 1:
        choice_data_pandas_training = SP_for_update
    if SP_RP_estimation == 1:
        temporarilytrainingRP = RP_for_update[RP_for_update.index.isin(choice_data_pandas_test.index)]
        choice_data_pandas_training = pd.concat([SP_for_update,temporarilytrainingRP])

    #Transform data into Biogeme format
    database_training = db.Database("database_training", choice_data_pandas_training)
    database_test = db.Database("database_test", choice_data_pandas_test)

    #Define the name of the variables as python variables
    globals().update(database_training.variables)

    #Define parameters to be estimated
    beta_a1t1 = Beta("beta_a1t1", 0, None, None, 0)
    beta_b1t1 = Beta("beta_b1t1", 0, None, None, 0)
    beta_c1t1 = Beta("beta_c1t1", 0, None, None, 0)
    beta_d1t1 = Beta("beta_d1t1", 0, None, None, 0)
    beta_e1t1 = Beta("beta_e1t1", 0, None, None, 0)
    beta_f1t1 = Beta("beta_f1t1", 0, None, None, 0)
    beta_g1t1 = Beta("beta_g1t1", 0, None, None, 0)
    beta_h1t1 = Beta("beta_h1t1", 0, None, None, 0)
    beta_i1t1 = Beta("beta_i1t1", 0, None, None, 0)
    beta_j1t1 = Beta("beta_j1t1", 0, None, None, 0)
    beta_k1t1 = Beta("beta_k1t1", 0, None, None, 0)
    beta_l1t1 = Beta("beta_l1t1", 0, None, None, 0)
    beta_m1t1 = Beta("beta_m1t1", 0, None, None, 0)
    beta_n1t1 = Beta("beta_n1t1", 0, None, None, 0)

    #Define te utility functions
    V0 = 0
    V1 = beta_a1t1 + beta_b1t1*b_A1t1 + beta_c1t1*c_A1t1 + beta_d1t1*d_A1t1 + beta_e1t1*e_A1t1 + beta_f1t1*f_A1t1 + beta_g1t1*g_A1t1 + beta_h1t1*h_A1t1 + beta_i1t1*i_A1t1 + beta_j1t1*j_A1t1 + beta_k1t1*k_A1t1 + beta_l1t1*l_A1t1 + beta_m1t1*m_A1t1 + beta_n1t1*n_A1t1
    # Associate utility functions with the numbering of alternatives
    V = (0: V0,
         1: V1)
    # Associate the availability conditions with the alternatives
    av = (0: 1,
          1: 1)

    #Definition of the model, contribution of each choice to the loglikelihood function.
    logprob_MNL_model1 = models.logit(V, av, CHOICE)
    # Create the Biogeme object
    biogeme_MNL_model1 = bio.BIOGEME(database_training, logprob_MNL_model1)
    biogeme_MNL_model1.modelName = "MNL_model1"
    # Estimate the parameters
    results_MNL_model1 = biogeme_MNL_model1.estimate()
    #Get beta's of trained model to validate
    betas = results_MNL_model1.getBetaValues()
    beta_list.append(betas)
    #Get and save results
    biogeme_MNL_model1.generateHtml = True
    biogeme_MNL_model1.generatePickle = True
    # Get the results in a pandas table
    results_MNL_model1_dataframe = results_MNL_model1.getEstimatedParameters()
    gs_MNL_model1 = results_MNL_model1.getGeneralStatistics()
    #Define probabilities
    prob_Niets = models.logit(V, av, 0)
    prob_Openemen = models.logit(V, av, 1)
    #Define simulation
    simulate = {'Prob. Niets': prob_Niets,
               'Prob. Openemen': prob_Openemen}
    #Define test simulation model
    biogeme_MNL_model1_test = bio.BIOGEME(database_test, simulate)
    biogeme_MNL_model1_test.modelName = "biogeme_MNL_model1_test"
    #Get validation results
    results_test_MNL_model1 = biogeme_MNL_model1_test.simulate(theBetaValues=betas)
    #Get highest probability for every test record
    prob_max_MNL_model1 = results_test_MNL_model1.idxmax(axis=1)
    prob_max_MNL_model1 = prob_max_MNL_model1.replace({'Prob. Niets': 0, 'Prob. Openemen': 1})
    #Get accuracy and add to list
    data_forCM = {'y_Actual': choice_data_pandas_test['CHOICE'], 'y_Predicted': prob_max_MNL_model1}
    df_forCM = pd.DataFrame(data_forCM, columns=['y_Actual', 'y_Predicted'])
    confusion_matrix = pd.crosstab(df_forCM['y_Actual'], df_forCM['y_Predicted'], rownames=
    print(confusion_matrix)
    trues = sum(df_forCM["y_Actual"] == df_forCM["y_Predicted"])
    accuracy = trues/confusion_matrix.to_numpy().sum()
    print('Global accuracy (e.g. correspondence) of the model:', accuracy)
    accuracy_list.append(accuracy)
```

Figure G.3: The code to estimate and validate a recommendation generator model for an update with the choices stored in in the temporary choice bases. Required for implementation of all components in table G.1.

User settings to specify choice weight of experiment and real-life choices

```
In [7]: #Frontend: Choose number of iterations/folds
k = 10

#Frontend: Select data set to estimate parameters
SP_estimation = 0
SP_RP_estimation = 1

#Frontend: Specify multiplication

#influence_specification = input("please specify importance rating") #can also be done with open question instead of sl

def influence(Exp_RealLife):
    return Exp_RealLife

influence_specification = interactive(influence, Exp_RealLife=widgets.IntSlider(min=-10, max=10, step=1, value=1))
display(influence_specification)

Exp_RealLife 
```

Backend user settings: get data according to user settings

```
In [11]: #Backend: Multiply data according to balance slider

if -10<= influence_specification.result <=-1:
    multiplied_SP = pd.concat([SP]*(abs(influence_specification.result)))
    multiplied_RP = RP
elif 1<= influence_specification.result <=10:
    multiplied_RP = pd.concat([RP]*influence_specification.result)
    multiplied_SP = SP
else:
    multiplied_SP = SP
    multiplied_RP = RP

SP_for_update = multiplied_SP
RP_for_update = multiplied_RP
print("Sample sizes SP observations:", len(multiplied_SP), "-", "Sample sizes RP observations:", len(multiplied_RP))

Sample sizes SP observations: 425 - Sample sizes RP observations: 154
```

Figure G.4: The code to adjust weight with which experiment choices (SP) and real-life (RP) choices are multiplied (front-end) and to store the multiplied choices in the temporary choice base (back-end). This code shows how an user can specify the weight with an importance rate (a multiplication of choices with a chosen integer). Required for implementation of component 2 in table G.1.

User settings to specify the balance between experiment and real-life choices

```
In [7]: #Frontend: Choose number of iterations/folds
k = 10

#Frontend: Select data set to estimate parameters
SP_estimation = 0
SP_RP_estimation = 1

#Frontend: Specify multiplication
#influence_specification = input("please specify importance rating") #can also be done with open question instead of sl

def balance_slider(balance):
    return balance

balance_specification = interactive(balance_slider, balance=widgets.IntSlider(min=0, max=100, step=1, value=50))
display(balance_specification)

balance 
```

Backend user settings: get data according to user settings

```
In [9]: #Backend: Set sample sizes
SP_sample_size = len(SP)
RP_sample_size = len(RP)

#Backend: Multiply data according to balance slider
if SP_sample_size > RP_sample_size or (SP_sample_size == RP_sample_size and balance_specification.result<50):
    multiplier = 1
    RP_size_vergroter = SP_sample_size * (100-balance_specification.result)/balance_specification.result/RP_sample_size
    while not RP_size_vergroter.is_integer(): #je wil RP observaties alleen met geheel aantal vermenigvuldigen
        multiplier += 1
    #print(multiplier)
    RP_size_vergroter = SP_sample_size * (100-balance_specification.result)/balance_specification.result/RP_sample_size
    #print(RP_size_vergroter)
    balanced_SP = pd.concat([SP]*multiplier) #SP vregroten om aantal keer verm
    balanced_RP = pd.concat([RP]*int(RP_size_vergroter))
elif RP_sample_size > SP_sample_size or (SP_sample_size == RP_sample_size and balance_specification.result>50):
    multiplier = 1
    SP_size_vergroter = RP_sample_size*balance_specification.result/(100-balance_specification.result)/SP_sample_size
    while not SP_size_vergroter.is_integer():
        multiplier += 1
    SP_size_vergroter = RP_sample_size *balance_specification.result/(100-balance_specification.result)/SP_sample_size
    #print(RP_size_vergroter)
    balanced_RP = pd.concat([RP]*multiplier)
    balanced_SP = pd.concat([SP]*int(SP_size_vergroter))

SP_for_update = balanced_SP
RP_for_update = balanced_RP

print("For the coming update, the balance is", balance_specification.result, "% such SP set contains", len(SP_for_update))

For the coming update, the balance is 70 % such SP set contains 229075 observations. The RP set contains 98175 observations.
```

Figure G.5: The code to adjust weight with which experiment (SP) and real-life (RP) choices are multiplied (front-end) and to store the multiplied choices in the temporary choice base (back-end). This code shows how an user can specify the weight with an importance balance (a multiplication of choices with a balance slider). Required for implementation of component 2 in table G.1.

User settings to choose between exclusion or weighting of a real-life choice sub group

```
In [6]: #Frontend: Choose number of iterations/folds
k = 10

#Frontend: Select data set to estimate parameters
SP_estimation = 0
SP_RP_estimation = 1

#Frontend: Select exclusion OR RP specific multiplication or both
RP_specific_exclusion = 0
RP_specific_multiplication = 1

#Frontend: Specify multiplication
def influenceRP(WeightSubgroup):
    return WeightSubgroup
if RP_specific_multiplication == 1:
    influenceRP_specification = Interactive(influenceRP, WeightSubgroup=widgets.IntSlider(min=-10, max=10, step=1, display=influenceRP_specification))
#influence_specification = input("please specify importance rating") #can also be done with open question instead

WeightSub... 
```

User settings to make sub groups

```
In [7]: #Frontend: Characteristic menu: specify based on which characteristics choices should be excluded or multiplied
advice_characteristic = 0
date_characteristic = 0
time_characteristic = 0
confidence_characteristic = 1
deviation_characteristic = 0
expertise_characteristic = 0
discipline_characteristic = 0

#Frontend: Exclusion criteria menu: specify which choices should be excluded from the learning
advice_criterion = 50 #if BAIT was less than X percent sure
date_criterion_begin = '2020-10-20'
date_criterion_end = '2020-12-20'
time_criterion_begin = '09:00:00'
time_criterion_end = '17:00:00'
confidence_criterion = 60 #lower than X
deviation_criterion = 1 #if not deviating
expertise_criterion = 'junior' #only senior choices
discipline_criterion = ['c', 'd'] #only disciplines A,B
```

Figure G.6: The code to select a sub group of choices (for instance all choices made by juniors) and exclude these choices or multiply these choices with a weight (here an importance rate, alternative is importance balance). Required for implementation of component 3 and 4 in table G.1.

Backend user settings: get data according to user settings (exclusion/multiplication and the selected sub group)

```
In [14]: #Backend: Get the indices of the specified characteristics & criteria
index = RP_characteristics.index
indices_to_exclude = []
indices_to_exclude_date_begin = []
indices_to_exclude_date_end = []
indices_to_exclude_time_begin = []
indices_to_exclude_time_end = []
indices_to_exclude_confidence = []
indices_to_exclude_deviation = []
indices_to_exclude_expertise = []
indices_to_exclude_discipline = []

if date_characteristic == 1:
    condition_date_begin = RP_characteristics["Date"] <= date_criterion_begin
    indices_to_exclude_date_begin = index[condition_date_begin].tolist()

if date_characteristic == 1:
    condition_date_end = RP_characteristics["Date"] >= date_criterion_end
    indices_to_exclude_date_end = index[condition_date_end].tolist()

if time_characteristic == 1:
    condition_time_begin = RP_characteristics["Time"] <= time_criterion_begin
    indices_to_exclude_time_begin = index[condition_time_begin].tolist()

if time_characteristic == 1:
    condition_time_end = RP_characteristics["Time"] >= time_criterion_end
    indices_to_exclude_time_end = index[condition_time_end].tolist()

if confidence_characteristic == 1:
    condition_confidence = RP_characteristics["Confidence"] <= confidence_criterion
    indices_to_exclude_confidence = index[condition_confidence].tolist()

if deviation_characteristic == 1:
    condition_deviation = RP_characteristics["Deviation"] == deviation_criterion
    indices_to_exclude_deviation = index[condition_deviation].tolist()

if expertise_characteristic == 1:
    condition_expertise = RP_characteristics["Expertise"] == expertise_criterion
    indices_to_exclude_expertise = index[condition_expertise].tolist()

if discipline_characteristic == 1:
    condition_discipline = RP_characteristics["Discipline"].isin(discipline_criterion)
    indices_to_exclude_discipline = index[condition_discipline].tolist()

#Backend: Combine extracted indices
indices_to_exclude = indices_to_exclude_confidence + indices_to_exclude_deviation + indices_to_exclude_expertise
RP_to_include_characteristics = RP_characteristics.index.isin(indices_to_exclude), :)
RP_selection_to_include = RP.iloc[~RP.index.isin(indices_to_exclude), :]

In [15]: #RP specific multiplication
multiplied_specified_RP = pd.concat([RP_selection_to_include]*influenceRP_specification.result)
RP_not_multiplied = RP.iloc[~RP.index.isin(multiplied_specified_RP.index), :]
RP_with_multiplied_specification = pd.concat([multiplied_specified_RP, RP_not_multiplied]) #verduubelde observaties

In [17]: #Hieronder een plek om sonder alle instellingen aan te passen de oorspronkelijke/gespecificeerde data set wel of niet te
if advice_characteristic == 1 or confidence_characteristic == 1 or deviation_characteristic == 1 or expertise_characteristic == 1:
    if RP_specific_exclusion == 1:
        RP_for_update = RP_selection_to_include
        print("The temporary choice base is filled by excluding RP observations. The sample size is:", len(RP_for_update))
    if RP_specific_multiplication == 1:
        RP_for_update = RP_with_multiplied_specification
        print("The temporary choice base is filled by multiplying RP observations. The sample size is:", len(RP_for_update))
SP_for_update = SP

The temporary choice base is filled by multiplying RP observations. The sample size is: 185 .
```

Figure G.7: The code to create the temporary choice bases for an update according to the specifications of the clinical end user (see fig. G.6). Required for implementation of components 3 and 4 in table G.1.

G.2 Outcomes of executing scripts

The scripts were executed with hybrid choice data (see appendix G.1), meaning that the data was enriched with dummy data. As such, it is meaningless to make and present claims about the values resulting from the execution. However, the outcomes do expose the functionalities of the components defined by the architecture. Therefore, the outcomes of the scripts presented in appendix G.1) were discussed with Council to judge whether the components generate the intended functionalities. The main outcomes are shown in table G.2, table G.3, and table G.4. A brief guiding explanation of the outcomes is provided below.

- In table G.2 the parameter estimations for each choice attribute are presented that result from executing component 1 (see table G.1). A parameter in a Discrete Choice Modeling (DCM) model represents the relative importance assigned by choice makers of the attribute it was estimated for. To make the presentation more intuitive, the parameters were translated into a percentage indicating their relative importance. The first set of parameters are estimated based on experiment choices only. The second set of parameters is estimated based on both experiment choices and real-life choices. The current BAIT-based CDSS only estimates parameters based on experiment choices. The outcomes show that the parameters change according to the new real-life choices when real-life choices are included as is intended. When taking the new parameters as the new active model based on which the choice recommendation generator calculates choice recommendations, the CDSS is updated with the new knowledge in the decision-making context.
- In table G.3 the Correspondence rate is presented for updates with both experiment choices joint with real-life choices, and experiment choices only found with executing component 1 and 2 (see table G.1). The correspondence rate is needed to assess the performance. The outcome shows that the correspondence rate was found as intended. Moreover, it shows that it changes when informing the update with choices made in the real-life context. When the experiment data is assigned with a much larger weight, the performance does not exceed the performance of a model with only real-life choices. This is in line with expectations, because in this scenario the role of real-life choices is negligible. Finally, when requesting the validation sets that the implementation used, it could be confirmed that only the most recent choices were used for the performance assessment.
- In table G.4 the parameter estimations for each choice attribute are presented that result from executing component 2 with different weighting specifications (see table G.1). The outcomes show that the parameters change according to the new real-life choices when a clinical end user specifies a weighting as is intended. Component 2 thus enables to retrieve different choice models based on which the choice recommendation generator calculates choice recommendations.
- For the execution of component 3 and 4 no parameter estimations are presented. It is already shown with the execution of component 1 and 2 that parameters change as is intended both when real-life choices are added to the choice base for an update, and when different weights are used. It was rather important that the temporary choice bases are constructed according the specifications of a clinical end user. The executing of component 3 and 4 showed that the temporary choice bases contain the choices that meet the selection criteria specified by the clinical end user.

Table G.2: *Parameters based on experiment choices, and on experiment choices and real-life choices (joint) with component 1 (see appendix G.1)*

Variable	Parameters based on experiment choices	Relative importance	Parameters based on real-life choices	Relative importance
Rehousing possibility (regional/national)	-0.419425	12.8 %	-0.508854	15.2%
ICU capacity	0.444133	13.6 %	0.376721	11.2%
Acute condition patient	-0.250557	7.7%	-0.286038	8.5%
Age	-0.027118	0.8%	-0.031446	0.9%
Comorbidity: Cognitive impairment	-0.011982	0.4%	-0.009210	0.3%
Comorbidity: Heart/veins	-0.006853	0.2%	-0.005796	0.2%
Comorbidity: Pulmonary	-0.004810	0.1%	-0.002015	0.1%
Comorbidity: Kidney	0.003269	0.1%	0.003883	0.1%
Comorbidity: Liver	-0.003483	0.1%	-0.002432	0.1%
Comorbidity: Immune System	-0.002621	0.1%	-0.001432	0.0%
Pattern COVID pneumonia	-0.381564	11.7%	-0.452723	13.5%
Body Mass Index	-0.041514	1.3%	0.004899	0.1%
Frailty	0.940355	28.7%	1.0	30.9%
Treatment preference of patient	0.733978	22.4%	0.629536	18.8%

Table G.3: *Correspondence rates for model estimation based on experiment choices and real-life choices*

Sample sizes	Correspondence rate	Correspondence rate experiment choices only (sample size)
Equally important	83.33% (n=425)	78.75%
Real-life choices 2 times more important than experiment choices	82.97%	76.25% (n=425)
Experiment choices 2 times more important than real-life choices	81.67%	77.08% (n=850)
Real-life choices 6 times more important than experiment choices	82.97%	74.58% (n=425)
Experiment choices 6 times more important than real-life choices	77.92%	78.75% (n=2550)

Table G.4: *Parameters estimated with component 1 and 2 (see appendix G.1) based on a temporary in which experiment choices and real-life choices were weighted (Exp. = experiment choices, RL=real-life choices).*

Variable	RL 2x important	RL 4x important	Exp. 2x important	Exp. 4x important
Rehousing possibility (regional/national)	-0.667894	-0.583053	-0.443486	-0.479233
ICU capacity	0.428654	0.369168	0.400253	0.421518
Acute condition patient	-0.238060	-0.403993	-0.274853	-0.275518
Age	-0.037397	-0.029842	-0.027140	-0.028427
Comorbidity: Cognitive impairment	-0.008115	-0.004676	-0.009767	-0.010901
Comorbidity: Heart/veins	-0.007079	-0.004636	-0.006064	-0.006263
Comorbidity: Pulmonary	-0.000612	0.004187	-0.002453	-0.003911
Comorbidity: Kidney	0.006051	0.002696	0.003508	0.004029
Comorbidity: Liver	-0.002098	0.000505	-0.002403	-0.003233
Comorbidity: Immune System	-0.004007	- 0.010224	-0.003190	-0.001562
Pattern COVID pneumonia	-0.213049	-0.397302	- 0.408632	-0.363808
Body Mass Index	-0.006144	-0.023082	0.006736	0.025301
Frailty	1.038650	0.991802	0.942378	0.973326
Treatment preference of patient	0.537457	0.673378	0.659552	0.679430

Appendix H

Architecture implementation: mock-ups

H.1 Mock-ups point of view from Councyl

This section presents mock-ups that show two screens that can be entered by Councyl. In fig. H.1 it is shown how Councyl can access the anonymous metrics of all updates of a CDSS. In fig. H.2 the screen to adjust the minimal number of clinical end users for the creation of a sub group.

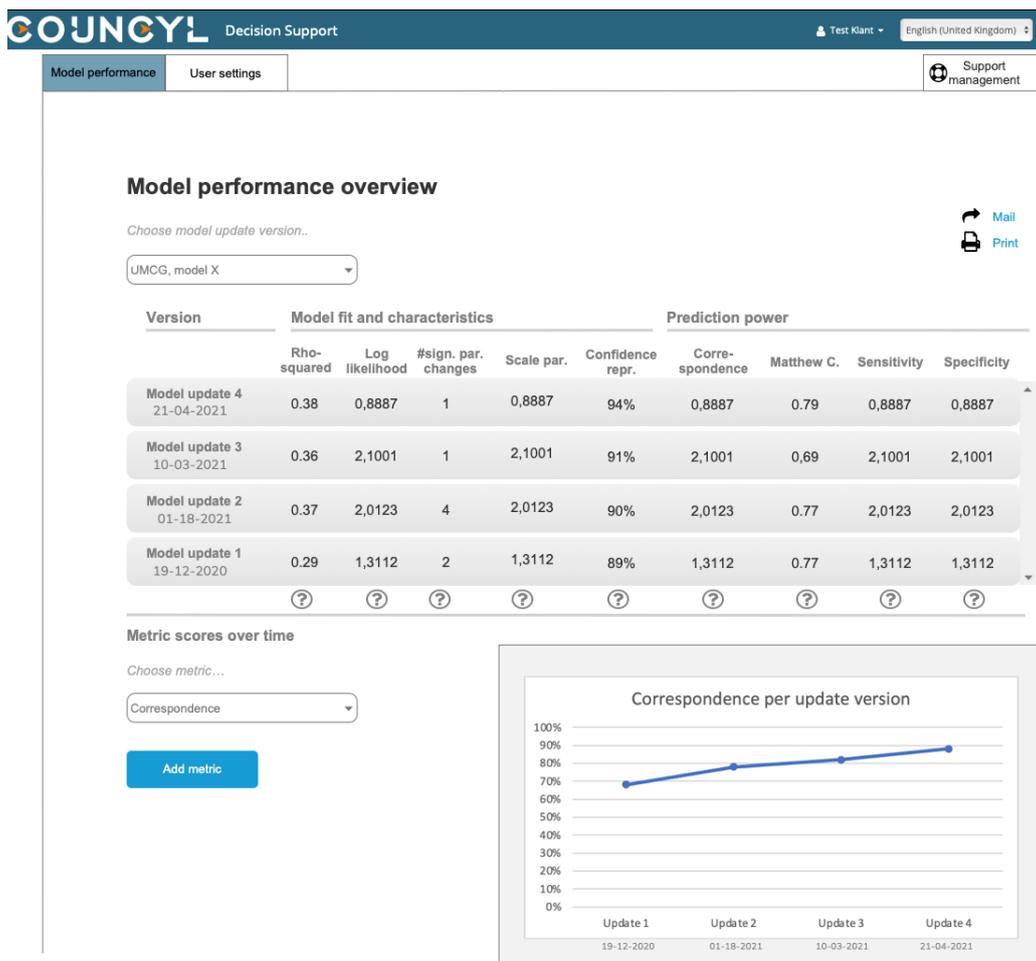


Figure H.1: Performance metrics overview for Councyl.

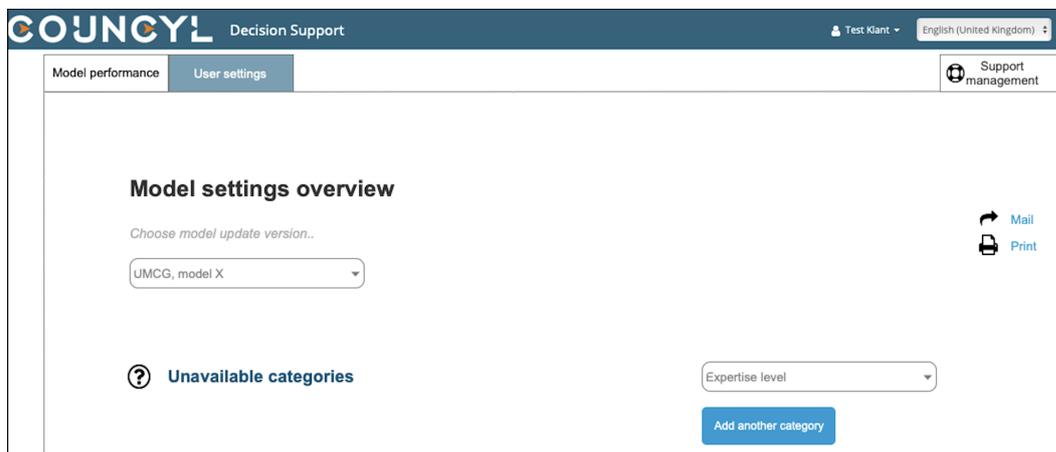


Figure H.2: Access to user settings for Councyl to make specific sub groups unavailable according to the Information Processing Agreement for privacy reasons.

H.2 Mock-ups point of view from clinical end users

This section presents the mock-ups of all architecture components from the point of view of Councyl. They are presented in a chronological order. This section thereby approaches the journey that a clinical end user could engage in when using a dynamic BAIT-based CDSS. The captions give a brief explanation of what each screen shows and what the main function of the screen is.

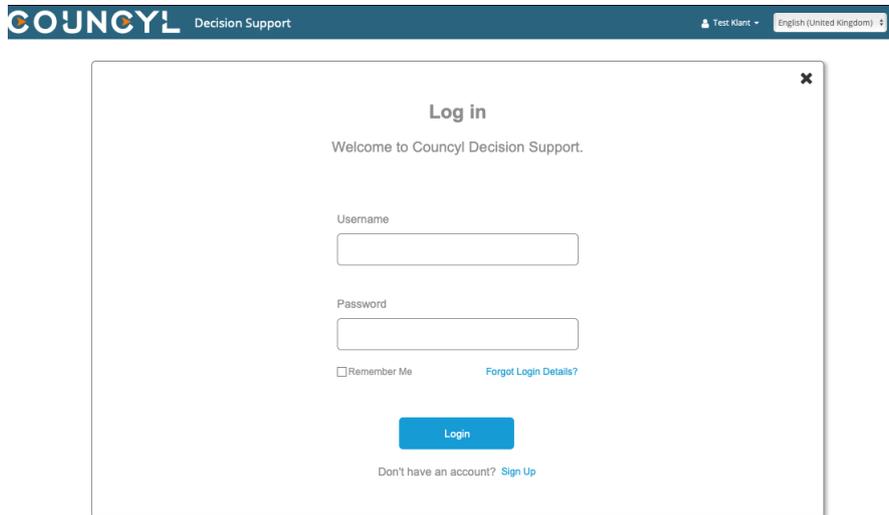


Figure H.3: The login screen for both Councyl and clinical end users.

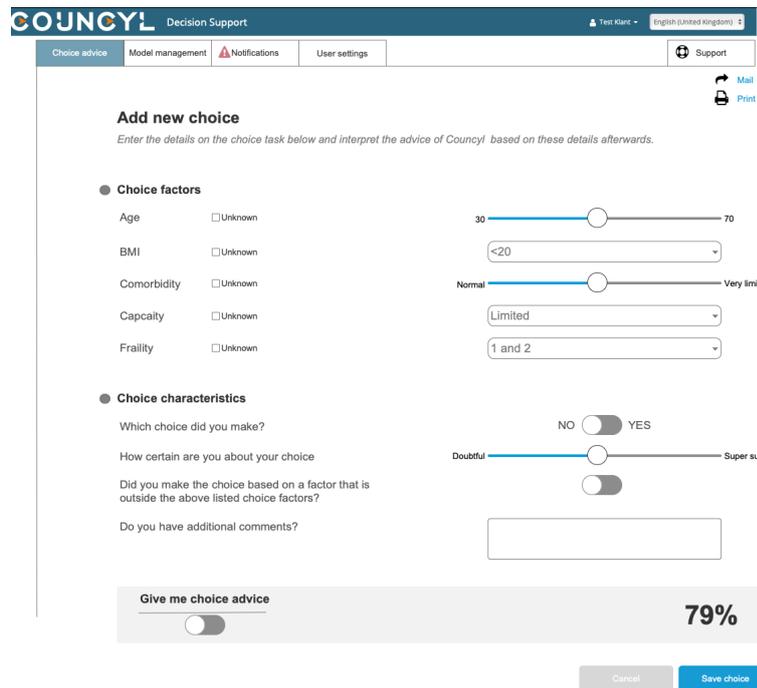


Figure H.4: The real-life choice storage and recommendation request screen. A clinical end user can enter the specification of a choice task he or she has to deals with. While filling in the choice specifications, the choice recommendation generator dynamically determines the recommendation. The end user can hide the recommendation if he or she does not want to be influenced in filling the choice details or in making a choice. Finally, the real-life choice specification can be stored.

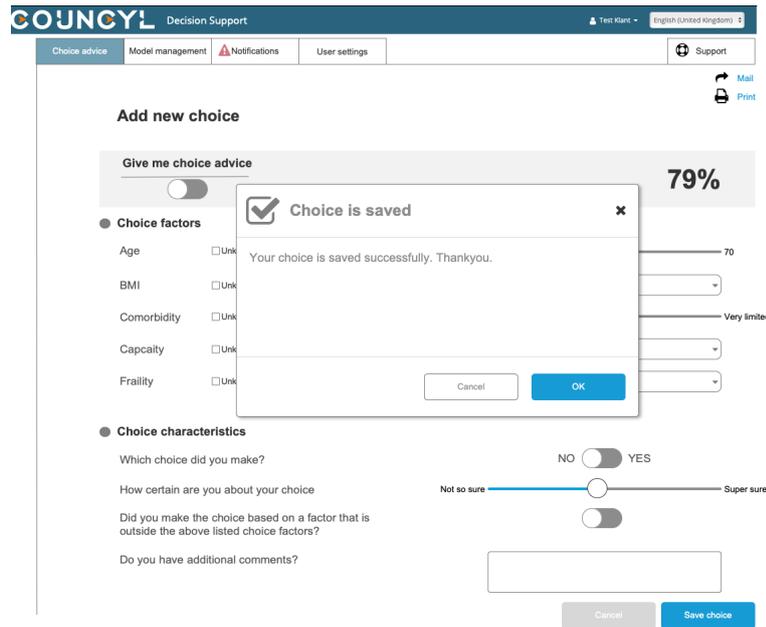


Figure H.5: A brief confirmation of a dynamic BAIT-based CDSS that the choice stored.

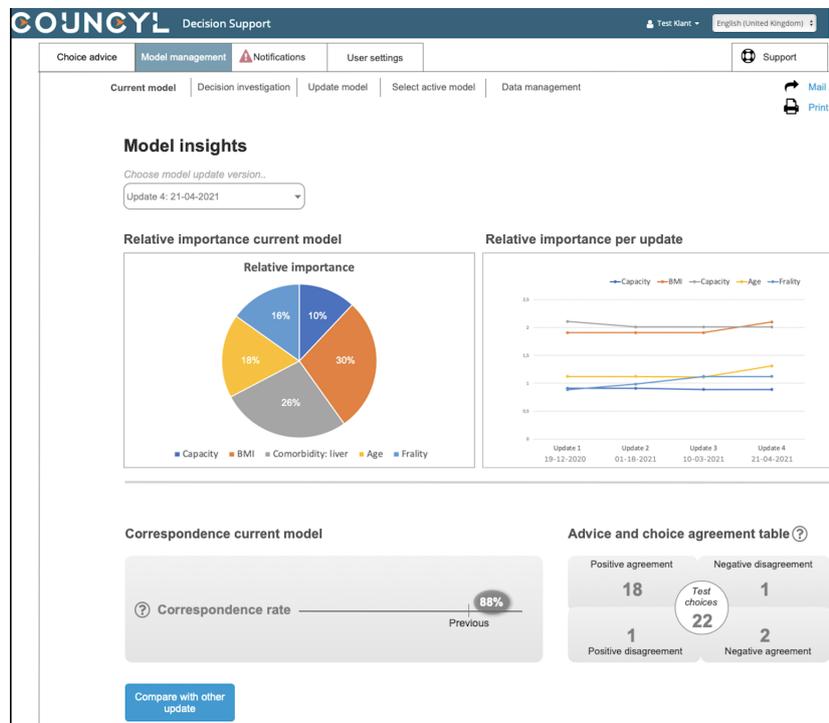


Figure H.6: The model insight screen shows the active model (in terms of the relative importance of all choice attributes for the decision-making) and the Correspondence rate the choice/clinical end user agreement table that were determined during the model validation process. Moreover, end users can request previous model updates to compare different model updates. Finally, it can be observed how the importance of the choice attributes changes over time (per update).

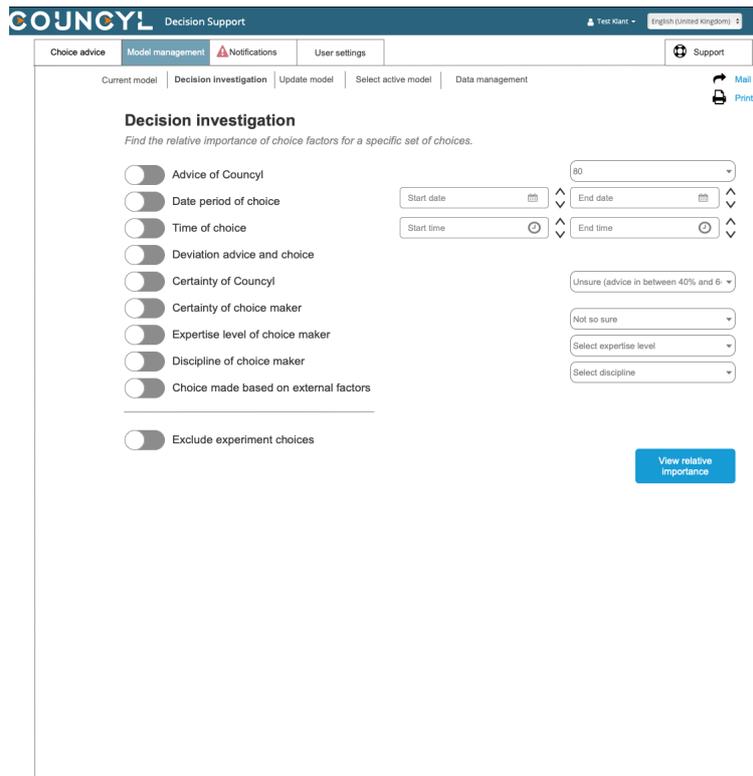


Figure H.7: The decision investigation allows clinical end users to specify a sub group of choices for which they want to see the model estimation and validation. For instance, the model for all choices made by senior clinical end users can be requested. The goal of this tool is either mutual learning or trying out model updates with different sets of choices.

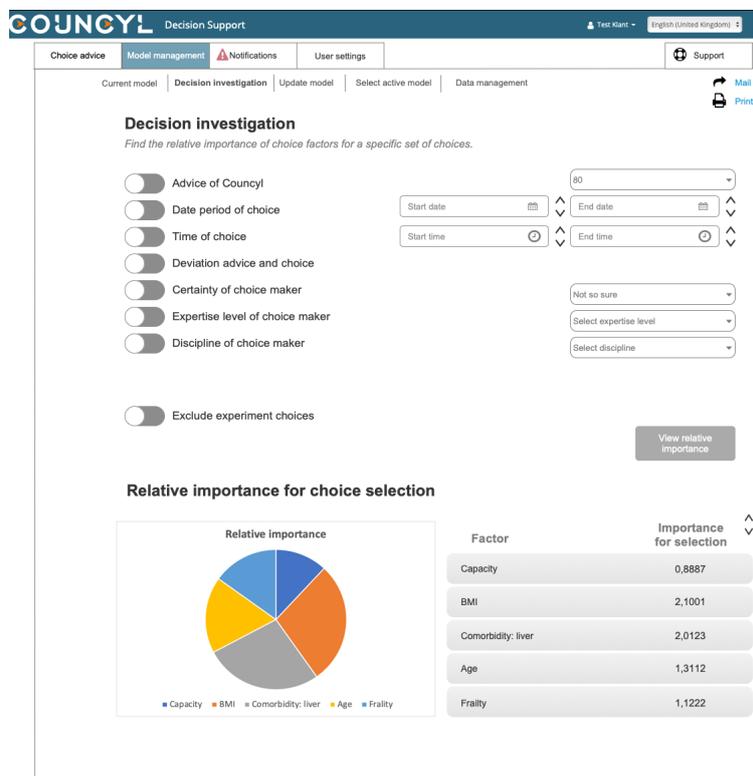


Figure H.8: As soon as a clinical end user has requested the model for a specific sub group, the model is directly and locally presented.

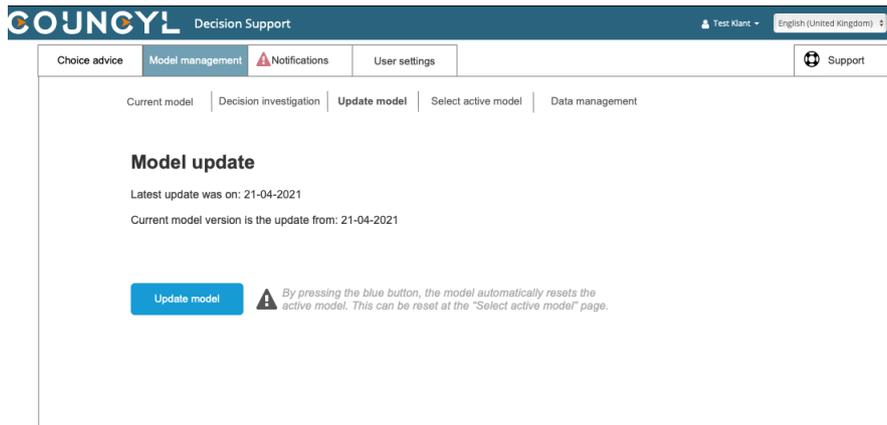


Figure H.9: The screen where clinical end users can activate a model update. The CDSS will directly update with the choices as specified in the user settings.

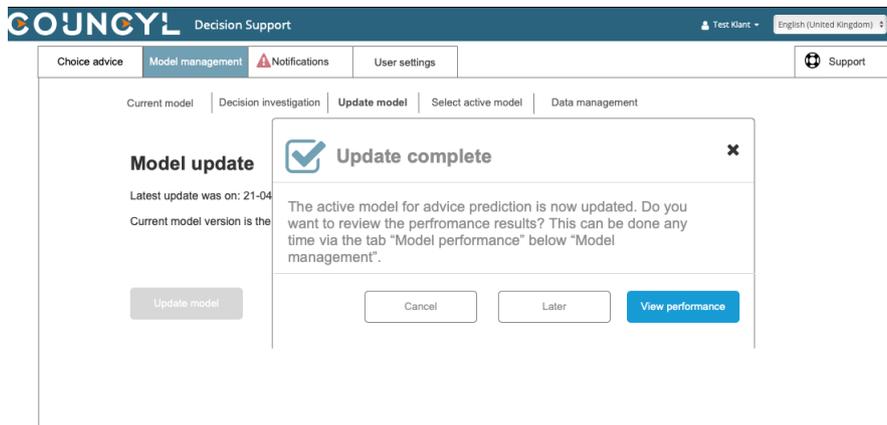


Figure H.10: The CDSS confirms an update, but does not directly presents the results of the update.

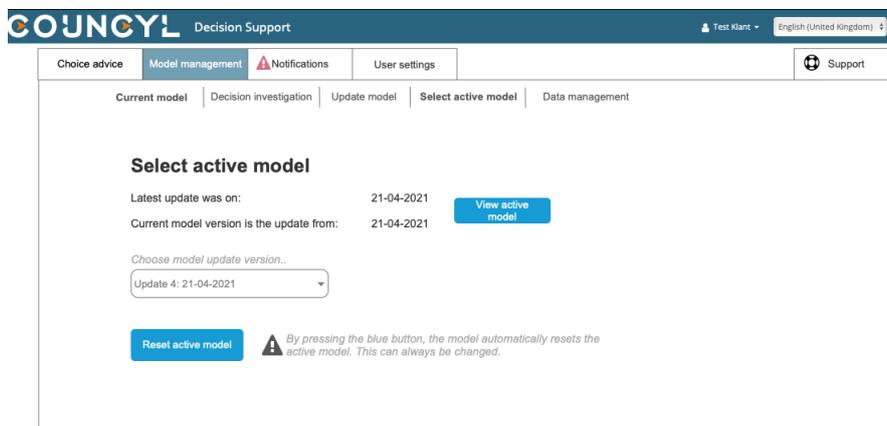


Figure H.11: The screen where clinical end users can reset the model that is used by the choice recommendation generator to provide choice recommendation. To this end, the end user can choose from all previous model updates.

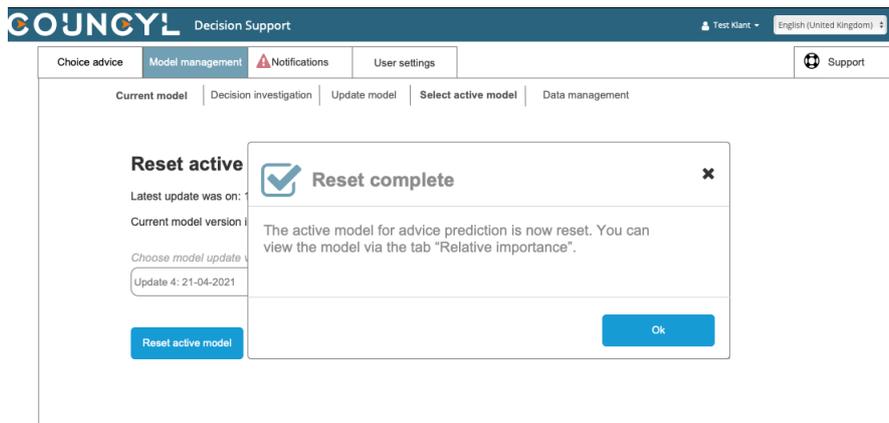


Figure H.12: The CDSS briefly confirms the selection of a new model for the choice recommendation generator.

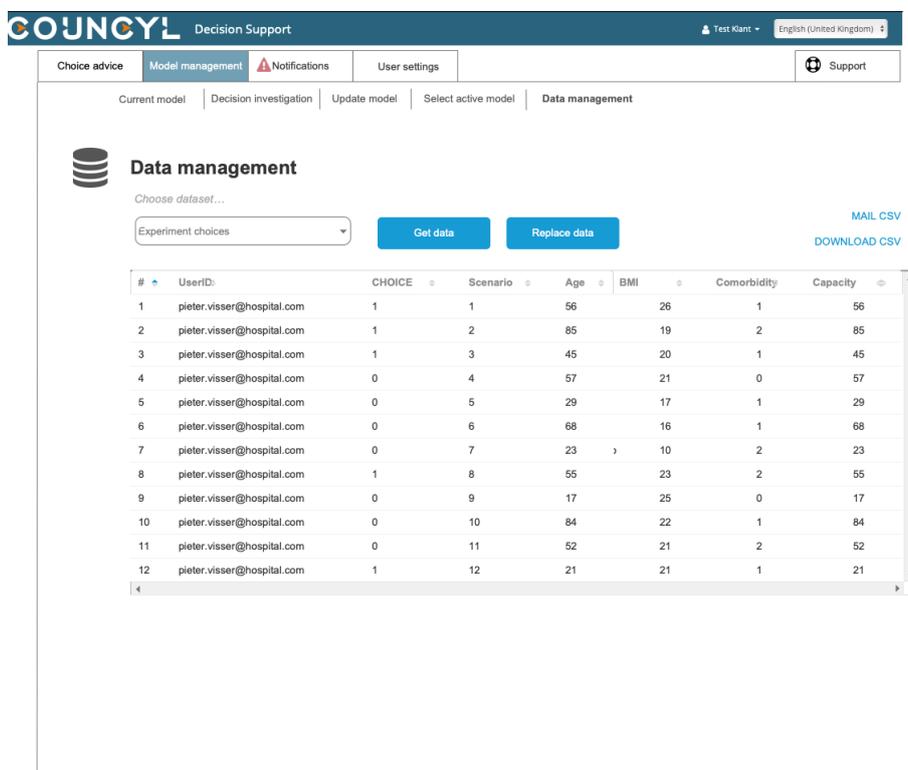


Figure H.13: A clinical end user can request all experiment choices and real-life choices he or she made in the past.

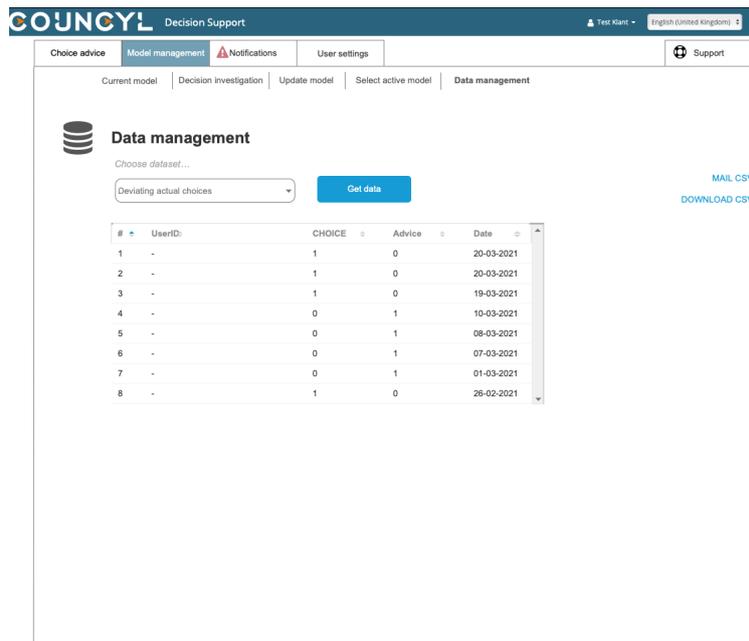


Figure H.14: The end user can access all the choices for which he or she deviated from the recommendation of the dynamic BAIT-based CDSS.

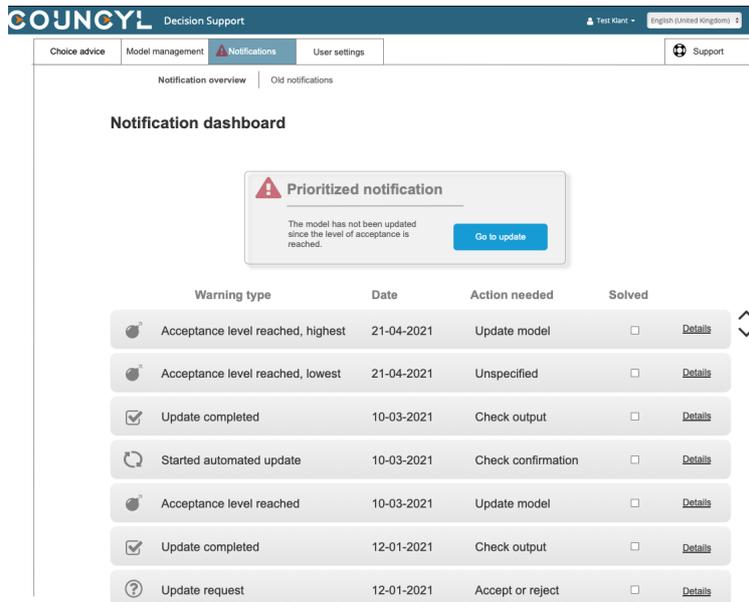


Figure H.15: The notification screen gives an overview of all warning feedback that is provided by the CDSS. The notification that warns that an update is needed remains visible until the end user updates the model.

COUNCYL Decision Support Test Klant English (United Kingdom)

Choice advice | Model management | Notifications | **User settings** | Support

Value preferences | Update preferences

Value preference settings

Level of acceptance Choose preferred value below...

1 Choose preferred action...

6 Send alert to authorized user

Request update **Extension 1A**

Repeated update

Every ... new real-life choices

Every time period **Extension 1B: automated update**

Majority threshold Choose preferred value below...

Number of validation iterations

Figure H.16: The user setting for the preference values show all options from which end users can choice to customize the CDSS towards the particular preferences. Most importantly, a clinical end user can choose here for the desired level of updating automation.

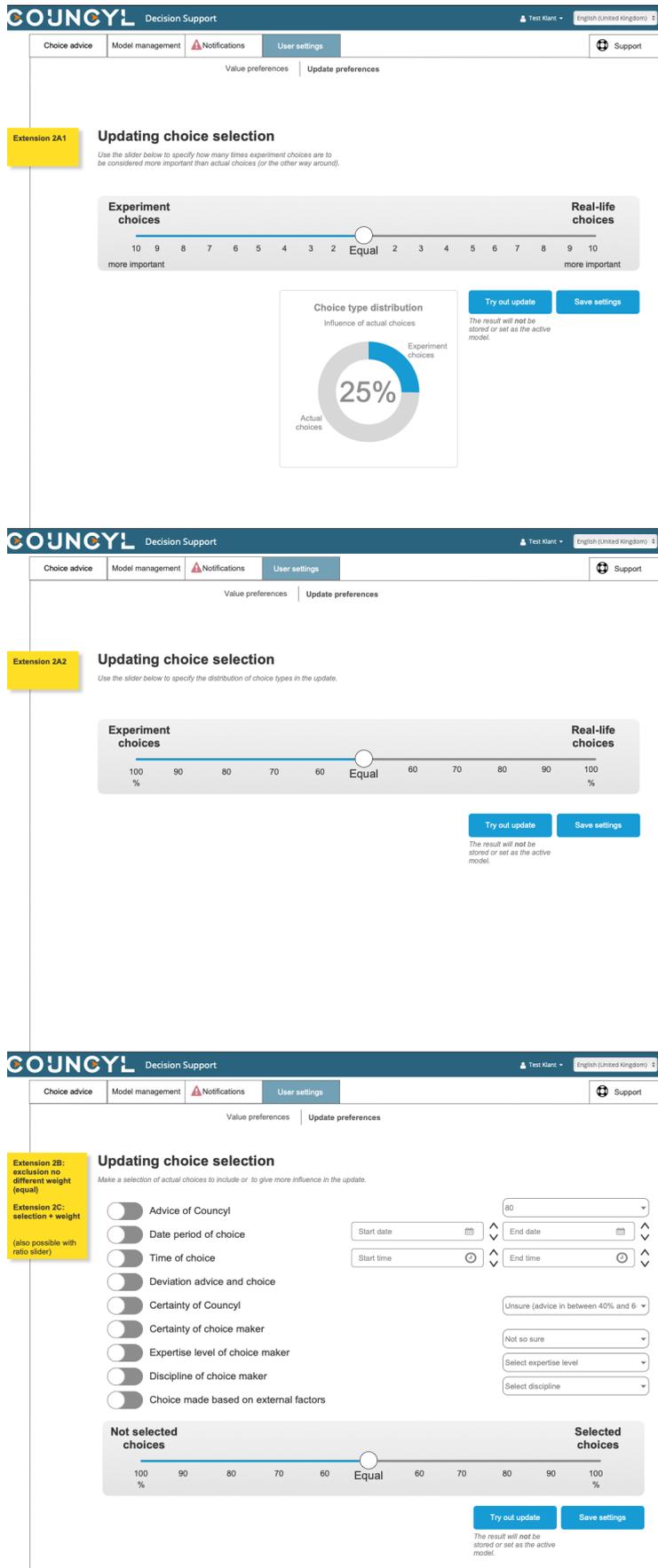


Figure H.17: One or more of these screens are only accessible if the clinical end user chose Information specification Extension 1, 2 or 3. If implemented, these screens allow clinical end users to specify how much specific types of choices will influence the model updates. To specify the weight, the end user can choose between an importance rate or an importance balance. All settings can be tried out, meaning that the results will be presented but will not be stored or influence the model that the CDSS uses for the generation of choice recommendation.

Appendix I

Organizational architecture implementation guidelines for Council

This section presents the main implementation guidelines associated to the architecture of a dynamic BAIT-based CDSS.

Targeted owner of guidelines. The guidelines aim to inform Council as provider of a dynamic BAIT-based CDSS. A provider is an entity that wants to use the architecture to develop BAIT-based CDSSs to serve varying healthcare decision-making contexts with a dynamic BAIT-based CDSS, and with the service that is associated to a dynamic BAIT-based CDSS.

Motivation of architecture implementation guidelines. The architecture defines the expansion of an existing support service. This does not only involve technological actions, but also poses organizational implications. As such, it cannot be perceived isolated from new organizational activities and challenges. With regard to these implications decisions are to be made.

Organizational architecture implementation guidelines. The guidelines give a formulation of the organizational activities needed from the CDSS provider and a procedure that guides these organizational activities. The activities are divided into three categories.

1. Organizational architecture implementation guidelines - preconditions. A decision on the ownership of the architecture should be made. This ownership involves the responsibility of keeping the knowledge captured by the architecture and the additional materials that enhance the understandability of the architecture at the right place such the right person(s) can use it in the way that was intended. In addition, the owner is concerned with the maintenance of the architecture. Although the architecture is adaptable over time, it does not change by itself. It requires an owner who keeps track of changing preferences of end users and other factors related to components of the architecture and adjusts the architecture accordingly.

2. Organizational architecture implementation guidelines - pre-implementation. The guidelines below define the actions that Council should perform together with the client to prepare a proper implementation of a dynamic-BAIT based CDSS.

- Conduct a meeting with main clinical end user for the specification of the user settings. At least three aspects need to be clarified and/or determined.
 1. Discuss all optional Information specification Extensions and Activation extensions, make the client aware of the consequences of all extensions with examples, and determine together which extensions suit the goals of the client with regard to the decision support.
 2. Go through all user settings variables, discuss what they mean, and together determine the initial set of values.
 3. Schedule a next meeting in which the experiences with the settings are discussed and potentially change the initial settings.

- Inform the client on the information privacy considerations and give the client time to process the considerations. Consequently, conduct a meeting with the main end user to fill in an Information Processing Agreement. At least the following aspects should be covered:
 - The minimal size of clinical end users needed before a sub group can be made accessible. The lower bound is always two, because information cannot be retrievable to individuals.
 - The information they want to share with Council. This can either be nothing, all data (including decision-making behaviour information (relative importance) as well as model fit and performance data) or only anonymous data (performance assessment). If the first option is chosen, the information flow from the model update database to Council should be ignored.
- Inform the client on the information a dynamic BAIT-based CDSS generates and make the client aware of the meaning and implications of this information. For instance, that the information can show what group of clinical end users are often not confident about their decision-making or is often deviating from the expected way of reasoning. Council needs to walk through the pieces of information the CDSS provides and discuss with the clinical end users what the power of the information is, but also what implications may result from having access to that information. Accordingly, Council should encourage the client to determine how this information should be processed, where in the organization it should be localized, who should be the owner of this information, and who should ensure the right actions are undertaken based on the information (see section 7.1.1).

3. *Organizational architecture implementation guidelines - post-implementation.* The guidelines below define the actions that Council should undertake as soon as a dynamic-BAIT based CDSS is implemented and is running in the client's decision-making context. The actions must be performed by Council in order to deliver the dynamic support service to the client as intended by the architecture.

- Both actions defined below are to be performed on an ongoing basis. Therefore, Council first needs to assign the responsibilities to employees such the actions will be performed.
- When the dynamic support service is running, additional support will be needed to guide on new service features. Moreover, different from the current static service, the dynamic service runs on a ongoing basis. The dynamic CDSS changes over time, resulting in a continuous flow of new insights and findings that are to be understood and interpreted correctly by clinical end users. clinical end users are expected to need additional support in doing so.
- If a clinical end user wants to replace the experiment choice base, Council should organize a new choice experiment and replace all experiment choices in the experiment choice base with the choices collected during the new experiment.
- The implemented dynamic CDSSs will generate large amounts of data that is of interest to Council. For instance, information about the performance of and the scale parameters estimated by CDSSs in different applications context. Council needs to process this information on a continuous basis. Moreover, Council needs to link this information with new organizational and technical process. For instance, ensuring there are consequences in place in case the situation is encountered in which multiple CDSSs show a decreasing performance or show a significant scale parameter that is too large to ignore. To allocate these important activities, someone has to be assigned with the role of taking care of the information and ensuring the necessary consequences or actions are .

Appendix J

Overview of the data objects defined by the architecture of a dynamic BAIT-based CDSS for Council

The data overview in fig. J.1 presents all data objects that the architecture of a dynamic BAIT-based CDSS as designed for Council needs to operate successfully and generate all functions defined in the architecture. The overview has four functions. The foremost functions of this overview is to keep the architecture clear: by defining all data objects in a separate document, they do not have to be visualized in the architecture. Second, because the overview shows the exact objects that are in place and provides a clear overview of the dependencies between the data objects and databases, it eases potential modifications in the information that the architecture components consume and produce. Moreover, because multiple versions of the overview will result from modifications over time, adjustments in the data objects can always be traced back later in time and there is always a back-up of the data organization in place. Third, because it emphasizes all important objects and shows the databases in which these objects are to be stored, Council can use the overview as a checklist when developing a BAIT-based CDSS. Fourth, it allows to specify the unities, and thereby to make an estimation of the size of the required data, in which the data objects need to be collected and stored, which is uncommon in and too detailed for an architecture design. Finally, it can be used to guide meetings with the client about the privacy of information and the preferences values in the user settings.

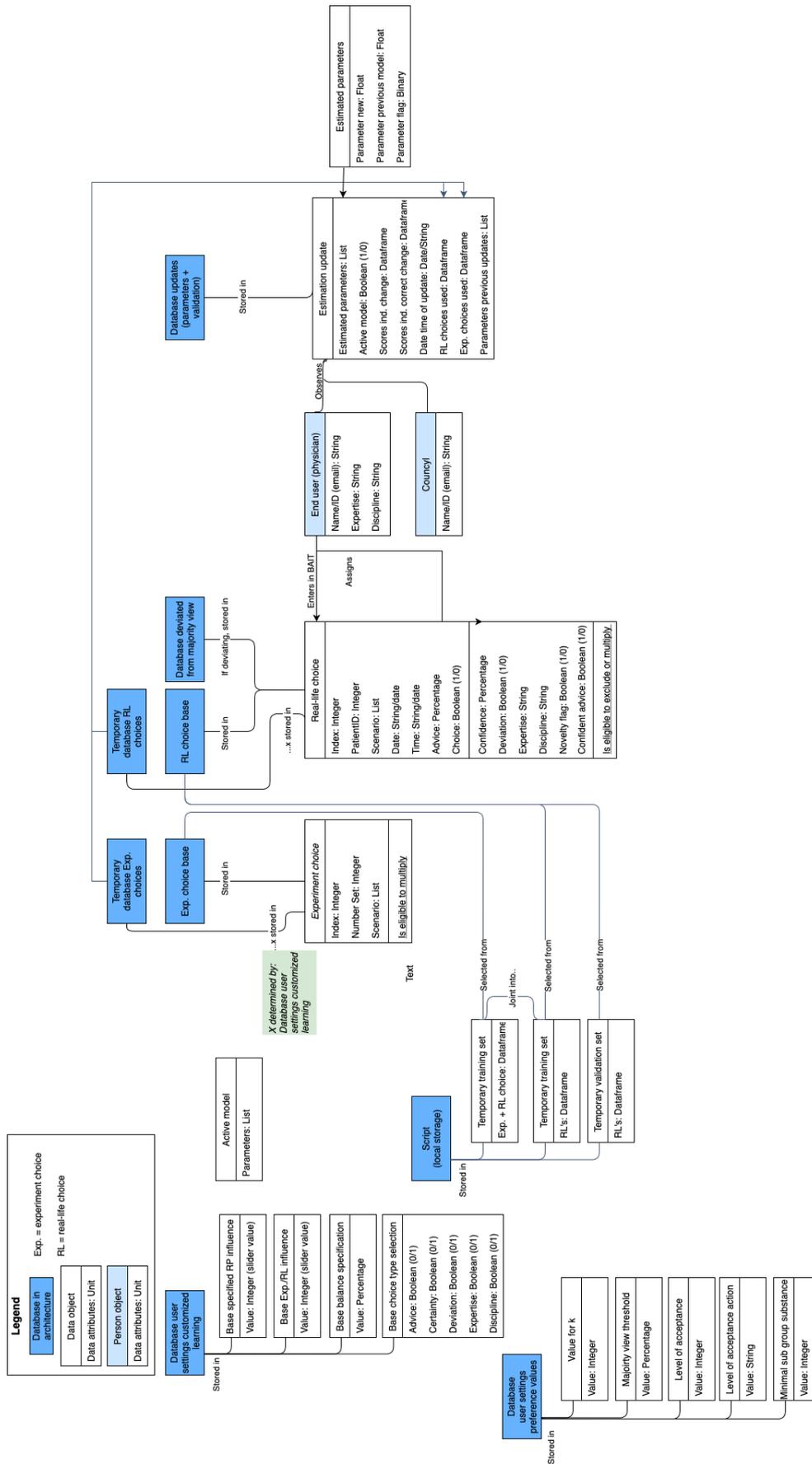


Figure J.1: Data overview architecture of a dynamic BAIT-based CDSS for Council.