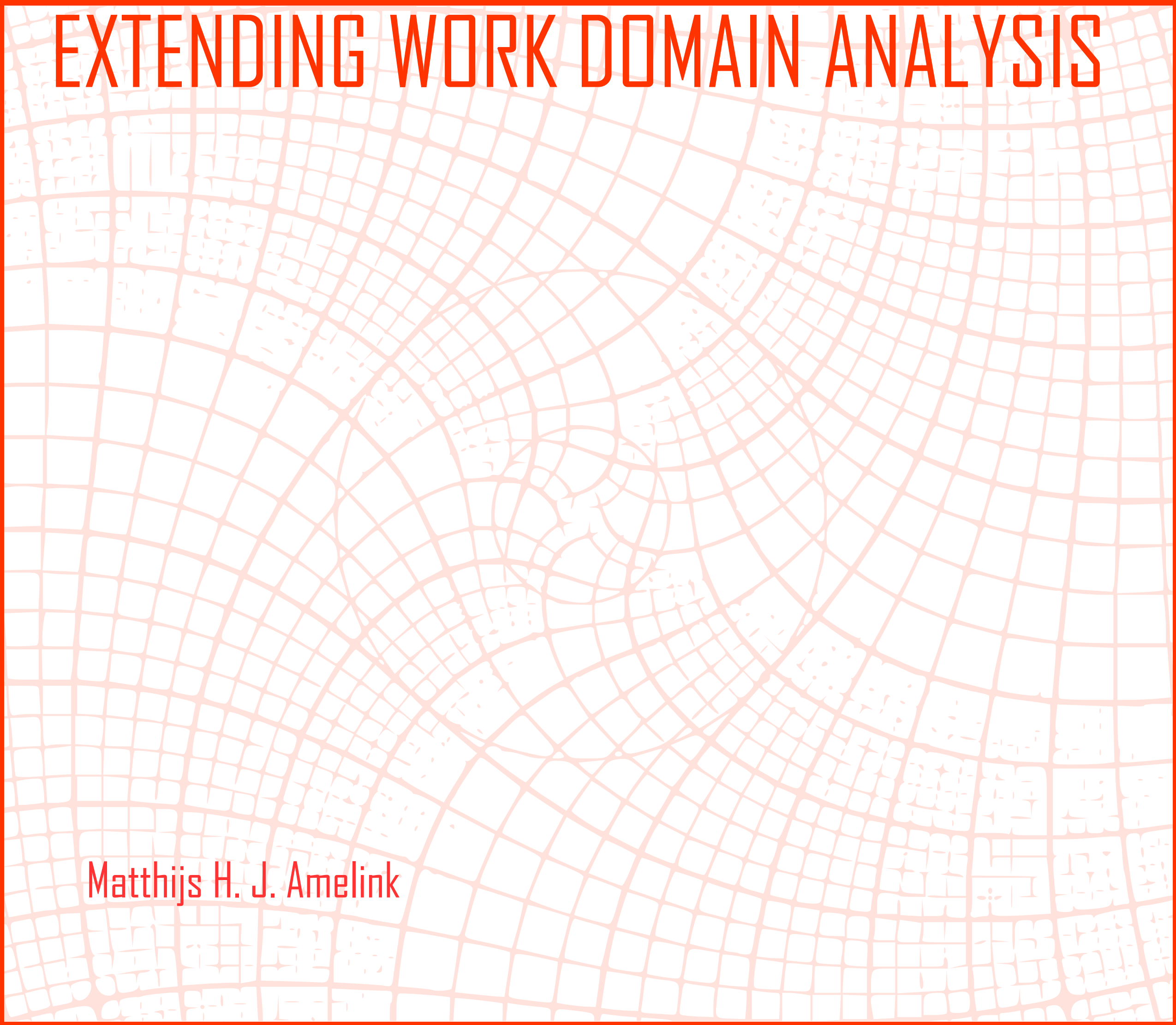
E A D

ECOLOGICAL AUTOMATION DESIGN, EXTENDING WORK DOMAIN ANALYSIS



Matthijs H. J. Amelink

Ecological Automation Design, Extending Work Domain Analysis

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof.ir. K.C.A.M. Luyben,
voorzitter van het College voor Promoties
in het openbaar te verdedigen op maandag 18 oktober 2010 om 12:30 uur
door

Matthijs Hendrik Jan AMELINK

vliegtuigbouwkundig ingenieur
geboren te Brummen

Dit proefschrift is goedgekeurd door de promotoren:
Prof.dr.ir. J.A. Mulder
Prof.dr.ir. M. Mulder

Samenstelling promotiecommissie:

Rector Magnificus	voorzitter
Prof.dr.ir. J.A. Mulder	Technische Universiteit Delft, promotor
Prof.dr.ir. M. Mulder	Technische Universiteit Delft, promotor
Dr.ir. M.M. van Paassen	Technische Universiteit Delft, copromotor
Prof.dr. J.M. Flach	Wright State University
Prof.dr.ir. P.J. Stappers	Technische Universiteit Delft
Dr. G.A. Jamieson	University of Toronto
Dr. C.H.M. Nieuwenhuis	Thales Research and Technology NL
Prof.dr. J. Dankelman	Technische Universiteit Delft, reservelid

The research described in this thesis was financially supported by Thales Nederland B.V. and by the Dutch Ministry of Economic Affairs as part of the BSIK–ICIS project, grant nr: BSIK03024.

Cover illustration inspired by "The grid of events" by M.C. Escher
Published and distributed by Matthijs H. J. Amelink
E-mail: matthijs@amelink.net

ISBN: 978-90-9025642-9

Summary

In high–risk domains like aviation, medicine and nuclear power plant control, automation has enabled new capabilities, increased the economy of operation and has greatly contributed to safety. However, automation increases the number of couplings in a system, which can inadvertently lead to more complexity from the perspective of the operator. The automation of a system transforms the work domain of the human operator, and his role changes from controlling the core processes to managing the automated processes. The complexity of the automation and the lack of proper support can make the control task's overall difficulty larger than it needs to be, restricting safety, productivity, and efficiency.

To address and limit the automation introduced complexity in the operator's work domain, and to find representations to support him, the *ecological approach to automation design* was taken. The ecological approach focuses on the relationship between the human operator and his work domain including the system he is controlling. The main research goals were to find how the ecological approach could be used to help limit the automation introduced complexity, and how the ecological approach could be used to support the human operator in controlling automated processes.

The formulation of Ecological Automation Design (EAD) was based on the Ecological Interface Design (EID) paradigm. One of the main underlying questions asked about the interface between the work domain and the human operator is: "how to represent work domain complexity?". The interface design paradigm was transformed into an automation design paradigm by first separating the automation component from the work domain and asking the same underlying question about the interfaces between the work domain, the human operator, and the automation. Then, the conceptual *shared domain representation* was defined to visualize that the apparent complexity of the system could be reduced when both the human operator and the automation view the same representation of constraints that the work domain imposes on control. As part of the ecological approach, Work Domain Analysis (WDA) was used to analyze and represent the constraints in a work domain. However, WDA is not yet fully developed and suffers from some methodological and conceptual issues. The research therefore, focused on the further development and extension of WDA to include the representation of automated processes. Four case studies were conducted, and each case study generated new insights into the application of and extension of WDA.

In the first case study, EID was applied to the design of the Energy Augmented Tunnel In the Sky display. This display was designed to aid a pilot to fly the approach to landing by presenting energy management information. The WDA revealed the significance of the energy coupling between vertical flight path and speed control as an intermediate control goal. Based on the analysis, a creative design process resulted in a novel display that has the energy representations fully, and graphically integrated in the tunnel in the sky display. A preliminary evaluation indicated that the additional energy management information shown in relation to the control actions and control goals helped pilots to fly the approaches. The display is not expected to give a performance increase but to change the way in which pilots control the throttle and elevator to fly approaches.

The second case study was the analysis of the already existing Total Energy Control System (TECS). TECS is an unconventional automated flight control system that was based on the same energy management constraints as that were represented in the energy augmented display of the first case study. The design of TECS was mapped onto the abstraction hierarchy to represent the energy management principles as part of the whole automated system. The analysis and useful representation of TECS using the abstraction hierarchy was not straightforward. It involved a search for the interpretation of the levels of the abstraction hierarchy and the use of the means–ends relationship in conjunction with the aggregation relationship. The resulting WDA showed that the abstraction hierarchy could be used to

map out the reasons for TECS's design features. Many constraints were represented in the same space, which cluttered the energy management principles. The focus was put on the energy management principles through selective aggregation of the represented functions, but other design principles were omitted. To provide a complete representation of the system but without the clutter, the levels of control sophistication were introduced to represented nested control problems separately. At each level of control sophistication the abstraction hierarchy was applied, resulting in the Abstraction–Sophistication Analysis (ASA).

In the third case study, the ASA framework was used to guide the design of *SmartUAV*. SmartUAV is a newly designed mini–UAV system that is capable of controlling multiple small UAVs from a laptop computer. By designing and developing SmartUAV we gained hands–on experience with how WDA, and especially the ASA, helped to keep track of and deal with the automation introduced constraints in the design phase. The levels of control sophistication were used from the beginning to separate the different control problems in the domain. They ranged from flying the platform to the achievement of missions. Starting at the lowest level of control sophistication, each higher level allowed the designer to include a larger part of the complete work domain incrementally, and to focus on more sophisticated control of the UAV. Furthermore, the ASA supported the visualization of how automation transformed the work domain, thus how automated functionalities that were created at lower levels of control sophistication affected the (automated) functions at higher levels of control sophistication. This study showed that the ASA could span a much larger problem space than the original WDA through the nesting of abstraction hierarchies. The ASA provided a systematic way to address the abstraction of the control problems (levels of control sophistication) and the abstraction of functions per control problem (abstraction hierarchy).

The fourth case study dealt with the analysis of a subset of a well structured domain that lacks automation; sailboat racing. This study generated a clearer view on the nested structure that is inherent in a work domain, as apposed to the nested structure of the automation as found in TECS and SmartUAV. The nested structure inherent to this work domain was found to be the result of how sailboat racing has evolved over time, based on the capabilities of equipment, human performance and the racing rules. Due to the lack of automation, it became clear that human performance is in fact part of the work domain, in contrast to the original formulations of WDA.

The crew's performance formed the basis for achieving the more sophisticated control of boat speed, tactics and strategy, thus was essential in the analysis. It was shown that the performance of the human crew could be represented in the ASA at a level of control sophistication, while this could not be supported in a non–nested WDA based on a single abstraction hierarchy.

The four case studies exemplified WDA and led to its extension with a structure to explicitly nest abstraction hierarchies that map out different control problems: the ASA. Through generating the analyses, extensive modeling experience with the abstraction hierarchy was obtained, reducing its ambiguity and potential methodological and conceptual problems. We found that the abstraction hierarchy could be used to model the structure of the knowledge about a work domain but could not model the knowledge itself. Therefore, the abstraction hierarchy is a framework for structuring knowledge, linking different representations of a control problem, and explaining the reasons for design features of a system.

The abstraction hierarchy addressed the abstraction of elements belonging to a control problem, and the levels of control sophistication addressed the abstraction of the control problem itself. Representations in the ASA framework ranged from physical at the lower levels of control sophistication to non–physical at the higher levels of control sophistication. It allowed the structuring of, for example: the sailboat racing rules at the higher levels, and the law of conservation of energy at the lower levels. Although the application of the ASA did not inherently reduce the complexity of the design of SmartUAV, it enabled us to better understand the elements of the work domain that contribute to complexity of the system prior to and during its design. The extension of work domain analysis with the levels of control sophistication has led to a richer representation of the studied work domains than a single abstraction hierarchy or the abstraction–decomposition space.

Contents

1 Introduction

"The cockpit crew did not have information regarding the interrelationship between the (failure of the) left radio altimeter system and the operation of the autothrottle. Of all the available indications and warning signals, only a single indication referred to the incorrect autothrottle mode, namely the 'RETARD' annunciation on the primary flight displays. With the knowledge available to them at that time, the crew had no way of understanding the actual significance of these indications and warning signals and could not have been expected to determine the pending risk accurately." (The Dutch Safety Board, 2010).

– A Turkish Airlines Boeing 737-800 crashed during their approach to landing near Amsterdam on February 25th, 2009.

1.1 Automation trends

Automation has greatly affected the way we work today in many domains. It has improved safety, reduced cost, increased efficiency, enabled new capabilities, and reduced workload. In the aircraft cockpit, the work of pilots has changed tremendously over the past fifty years as automation was introduced. Their primary task has moved from stick and rudder control to that of supervisory control of many automated processes for flight control, flight management, and other functions. In these developments, safety has always dominated the agenda, and it has transformed air travel into one of the safest forms of travel that we know (Mulder, 2009).

The main trend is towards systems with a high degree of automated continuous control, with increasing automated decision making capabilities, and perhaps even with intelligence. In aviation, Unmanned Aerial Vehicles (UAVs) are currently the main platform to explore and further push the boundaries of automation. The demand for these systems is illustrated by the facts that in 2009 the US Air Force was training more pilots for UAVs than for manned aircraft, and in 2010, the Pentagon will acquire more unmanned aircraft than manned aircraft (Pincus, 2009; Brook, 2009). To be cost effective, one of the main requirements for UAV systems is the reduction of the number of operators per UAV. To reach that goal, these systems require a level of autonomy that can only be reached through very high levels of automation. For the casual observer, it may be counterintuitive that the desired high level of autonomy of these systems is one of the main challenges for human–machine interface design. Leveraging the human–machine combination can be the success factor in these systems, as it can open the doors to new capabilities and better and safer systems than could be achieved by fully autonomous systems only.

Another trend in automation is that the role of the human operator changes. The interactions of the human operator with his work domain change from the basic processes (e.g., control of flight) to the management of the automated systems. Besides knowledge of the primary processes in the work domain, the human operator must have more and more extended knowledge of how the automation works. Parasuraman, Sheridan, and Wickens (2000) present a model for types and levels of human interaction with automation where the role of the human operator changes with the increase of autonomy. Bainbridge (1982) suggests that the increased interest in human factors among engineers reflects the irony that the more advanced (or sophisticated) a control system is, the more crucial may be the contributions of the human in establishing safe operation.

The view on the co–functioning of the human operator and the automation is also subject to trends. Automation is framed as a team player to form a joint cognitive system with the human (Woods & Hollnagel, 2006; Hollnagel & Woods, 2005). Or, automated functions are framed as *agents* that are artificial entities that operate according to the goals that they have been given in collaboration with the human *actors*. This is envisioned by the *actor–agent–communities* (AAC) paradigm (Iacob, Nieuwenhuis, Wijngaards, Pavlin, &

van Veelen, 2009). In actor–agent–communities, operators (actors) and automation (agents) cooperate as a team to achieve common goals in the same work domain. In this paradigm, automation is seen as a separate entity from the system, and its position in the work domain is similar to that of the human operator.

1.2 Automation and human factors issues

Despite the clear benefits of automation, issues also arise. The change in the roles of the automation and the human operators brings a number of concerns about their performance. Bainbridge (1982) presents the adverse effects of automation as "ironies of automation". These are categorized into four main themes, which are extended with additional issues that fit the themes.

The first two ironies address issues that are associated with how the automation is designed. Irony three addresses the human's (in)abilities to perform certain tasks under the created circumstances. Irony four addresses the need to support the operator in his interactions with the automated system.

1.2.1 The first irony, shifting the source of error

The first irony occurs when systems are being automated. When human control is replaced by automated control, the possibilities for human error do not disappear, rather they shift from operation to system design. The system design phase, therefore, becomes the primary source of human intervention in the system and thus a primary source of human error (Stanton, Salmon, Jenkins, & Walker, 2010). This type of error typically involves an erroneous implementation of the designed processes that leads to improper functioning of the automation under certain circumstances.

1.2.2 The second irony, what cannot be automated

The second irony concerns the tasks that the human operator performs in these systems. Automation relieves the operator from some tasks but also creates new ones, the ones that cannot be automated. For example, ensuring that the complete system state remains within the boundaries of safe operation. The result is that "The operator can be left with an arbitrary collection of tasks and little thought may have been given to providing support for them." (Bainbridge, 1982).

When the human operator is eliminated from control of (parts of) the system by automation, the designers must foresee all possible (fault) situations and engineer solutions for them. This irony involves the occurrence of *unanticipated events* (Rasmussen, 1986). These are by definition events that the designers did not foresee happening and the system will not behave properly in such an event. Therefore, the operators are not explicitly supported to resolve the situation, while a resolution relies on the human operator's problem solving.

In these situations, the automation can in fact increase the complexity of the human operator's work domain inadvertently. Woods and Cook (1991) refer to this problem as *nosocomial automation* [1] They argue that "adding other individual devices, interface features, and capabilities appear to increase complexity very little but the environment is transformed gradually to one in which the overall complexity presents a barrier to successful practice under certain situations". This does not involve errors of the first irony, but the *proper* functioning of the automation itself becomes a problem that the operator has to deal with.

Mode confusion errors with automatic flight control systems illustrates this well. Sarter, Woods, and Billings (1997) discuss a number of aviation accidents that are directly related to the complexity of the flight control systems. The different behavior of the system in different control modes, the transitions between modes, and the interactions between the various parts of the system introduce complexity. The quote at the very beginning of this chapter is also an illustration of the invisible (and unknown) couplings in the system producing unpredictable results. The complexity, however, is not inherent to the work domain but resulted from the architecture of the system. Human factors issues with automation are not concerned only with improvement of the man–machine interface but address the entire automation architecture, and especially the logic by which the behavior of the complete system is defined.

[1]With this term Woods and Cook draw a parallel to a 'nosocomial infection' that is adversely contracted as a result of being hospitalized. The word 'nosocomial' is derived from Latin; nosocomi(um) meaning hospital.

1.2.3 The third irony, monitoring

The third irony concerns the ability of the human operator to intervene with the automated system that he is monitoring. This irony is also addressed by Sheridan (2000) in the *dilemmas* of humans in supervisory control. It is impossible for even a highly motivated human being to maintain effective visual attention towards a source of information on which very little happens, for more than about half an hour (Mackworth, 1950). Hence, the human operator cannot effectively carry out the task of monitoring a fully automated system and detect abnormal behavior.

Automation is very adept at relieving the human operator of tasks during routine situations, but is often least of help in difficult situations, which typically require the operator to take over control (Norman, 1990). As a supervisor, the human operator is out of the loop and his awareness of the system's state and evolution of the state is reduced. An operator's situation awareness is a result of predictions and decisions made during control. This information is not built up instantaneously, and the lack of it presents another barrier when the operator has to take over manual control.

Moreover, not being actively involved in control of the system prevents the operator from acquiring and maintaining the skills that are required in order to take over control in non–routine situations. By "automating the process, the human operator is given a task which is only possible for someone who is in online control" (Bainbridge, 1982). Due to automation the human operator loses situational awareness, experience and skills to take over control in non–routine situations, yet his confidence remains (M. R. Endsley, 1995).

To tackle this issue, the human operator can be explicitly made part of the control loop in routine situation, when he is actually redundant. Active participation provides safety benefits and allows the human operator to respond more flexible to unexpected events (Parasuraman et al., 2000; Sheridan, 2000). This, however, depends on the control task. For control tasks that do not require flexibility in decision making and with a low probability of system failure, higher levels of automation (human out of the loop) often provide the best solution (M. Endsley & Kaber, 1999).

1.2.4 The fourth irony, the impossible task

Qualified as a more serious irony by Bainbridge (1982), the automatic control system has been put in place because it does a better job than the human operator. Yet, the operator is asked to monitor it. Thus the operator needs to be able to check in real–time that the computer is executing the control tasks correctly.

First, the operator needs to know what 'proper operation' of the system actually means. Second, the operator needs to know what the proper system state is and how the system state should evolve over time. However, it is impossible for a human operator to check in real–time that the computer is correctly following its programmed rules. As Bainbridge (1982) points out, "one can therefore expect the operator to monitor the computer's decisions at some meta–level, to decide whether the computer's decisions are 'acceptable' ". This is related to the challenge of human–machine–interaction design and the issues that were addressed in the second irony.

1.3 Research goal

To be able to address the above issues, we start with analyzing the relationships between the operator, the work domain and the automation. This approach is the *ecological approach to automation design*. The term "ecological" is used in this thesis in the same way as it was coined by Vicente and Rasmussen (1992) in Ecological Interface Design (EID). It refers to the duality between the operator and the work domain that is very similar to the organism–environment reciprocity. This ecological approach to automation design is formulated as Ecological Automation Design (EAD), and is further detailed below in Section 1.4.

The distinction is made between a work domain's *inherent complexity*[2] and the *adverse complexity* that is added by the automation. The first is a property of the work domain and should not be simplified to preserve the relevant information. As Albert Einstein said: "Everything should be made as simple as possible, but not simpler" (Wikiquote, 2010). The second form of complexity is an addition to the work domain as the result of the designed automation. Automation increases the degree of coupling among part of a system and therefore increases the complexity (Woods, 1996). While it may not be possible to eliminate the adverse complexity, it should be minimized as much as possible through design.

[2]See Section 2.2.4 for the *notion of complexity* used in this thesis. In short: the elements in a work domain and their interrelations contribute to complexity.

1.3.1 Research questions

From the second irony, we focus on the problem that an operator is left with a set of arbitrary tasks that could not be automated. The automation transforms the work domain of the human operators, making the overall complexity and task difficulty larger than they need to be. To counter this issue, the transformation of the work domain should be guided in the design phase to reduce or eliminate the adverse complexity. Then:

Can the *ecological approach* be used to limit or eliminate the adverse complexity that is introduced by the automation in the design phase?

From the fourth irony, we focus on the problem of a human operator monitoring an automated system that is in some aspects better and faster than the human. The human operator should be supported on a 'meta–level' to monitor the system.

Can the *ecological approach* be used to support the human operator in monitoring and problem solving in highly automated systems?

Together, these questions address the challenges of human performance in complex (socio–)technical system early in the design phase. The problems discussed under the first irony, thus, the erroneous implementation of automation, are *not* addressed by this work. The problem of the third irony, thus, human performance under the current way of working with current automated systems is *not explicitly* addressed in this thesis either. The focus of this work lies on the bigger picture, it is a holistic view that transcends the problems that were created by the current automation of systems.

1.3.2 Contributions

These questions and the issues that were discussed above are not new. Reflecting on these problems by defining new terms and language to discuss them, does not seem to bring solutions. On the other hand, building interfaces and systems that bring partial improvements (in laboratory environments) does often not address the overall problem as presented. The problems are widely sketched, but concrete solutions and examples of their application to the design of real systems are much less common. For example, Hollnagel and Woods (2005); Woods and Hollnagel (2006) launched the term 'Joint Cognitive System' for the same problems discussed above but do not provide more than a collection of loosely coupled conceptual models. Concrete (examples of) design solutions are not proposed.

This research aims to contribute to the already vast body of knowledge on these problems by adopting a promising solution–oriented approach to real systems. Thereby, making the approach concrete and providing realistic and detailed examples. The generated insights are then used to sharpen the used analysis method.

1.4 Ecological automation design

The approach of Ecological Automation Design (EAD) is inspired on EID. To formulate the EAD approach, we start with examining the approach of EID and then include the automation component.

1.4.1 Ecological interface design

EID addresses the *core problems* of interface design. Figure 1.1(a) shows the structure of the interface design problem as presented by Vicente and Rasmussen (1992). The human operator and the actions he can take are considered separate from the work domain. The analysis of a work domain is independent of any actions that the human operator may take. The complex work domain and the human operator are connected through the interface. To come to a good interface design, two questions are asked by Vicente and Rasmussen (1992).

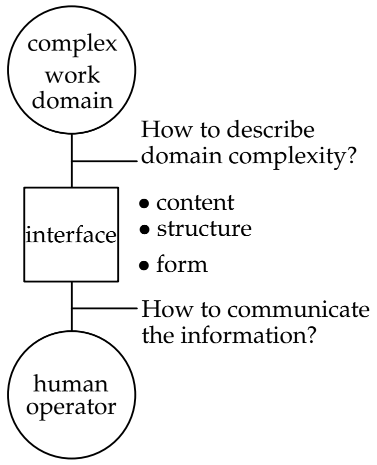
The first question is: "How to describe domain complexity?" This question relates to the characteristics of the work domain. A description of the relevant characteristics of the work domain is needed to determine the information *content* and *structure* of the interface. The information *content* represents the collection of elements in the work domain that is relevant to the control task for which the interface is designed. The information *structure* represents the interrelations of those elements that determine their behavior and how they constrain the control task. Vicente and Rasmussen (1992) propose the *abstraction hierarchy* as a formalism to describe work domain complexity: The abstraction hierarchy is "a psychologically relevant form of representing the constraints in a work domain in a way that allows operators to cope with unanticipated events".

The second question is: "How to communicate the information?". This question relates to the cognitive capabilities of the human operator. A model is needed that describes the various mechanisms that people have for processing information to determine the information *form*. The information form is the representation of the information content and structure in the interface towards the human operator. Vicente and Rasmussen (1992) propose the *Skills, Rules and Knowledge (SRK) taxonomy* as the model to answer this question. The information form of the interface should allow the human operator to process the information at the lowest level of cognitive processing that the task allows.

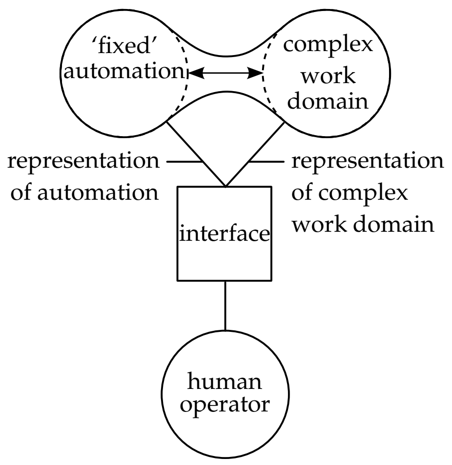
Both the abstraction hierarchy and the SRK taxonomy are adopted as axioms by Rasmussen and Vicente (1992). They are adopted as axioms in this thesis as well. Vicente's (2002) claim that EID has consistently improved performance compared with the industry state of the art, is taken as the basis for hypothesizing that similar improvements are possible for automation design. The focus of the ecological approach to automation design lies on the first question: "how to describe domain complexity?", but now for automation design instead of interface design.

1.4.2 Ecological approach to automation design

The definition of a work domain as given by Vicente (1999) excludes the human operator and all automation. His definition comes from the power plant control domain. By taking the theory into the vehicle control domain, we adapt this definition based on a realistic view of modern vehicle systems. Some automated functions should be considered as part of the work domain. For example: the computer controlled fuel injection system of a car, or the closed loop servo in a UAV's wing. These kinds of 'automated modules' are easily taken for granted because they represent more or less standard


complex work domain
How to describe domain complexity?
interface
• content
• structure
• form
How to communicate the information?
human operator

(a) The structure of the interface design problem, adapted from Vicente et al. (1992).


'fixed' automation
complex work domain
representation of automation
representation of complex work domain
interface
human operator

(b) The automation that is part of the work domain is shown as 'fixed' automation.

Figure 1.1: The relationship between the work domain, the automation, and the human operator from the perspective of the interface designer.

building blocks that are hidden from our attention. By including them in the work domain, their properties become part of the work domain description on which the higher level automation is based. If they are excluded from the work domain description, they are not taken into account and they become part of the design effort.

The designer should be aware of which automation belongs to the work domain and which automation is excluded from the work domain. Therefore, a distinction is made between *fixed automation* and the *to–be–designed automation*. The first is considered as part of the work domain, and forms the basis for further automation. Figure 1.1(b) shows the position of the fixed automation in the original structure of the interface design problem of Figure 1.1(a). The second is placed outside the work domain. Then, similar to the position of the human operator in the original structure of the interface design problem, an interface can be defined between the to–be–designed automation and the work domain. In this way, the work domain can be analyzed independently of any actions that the new automation may take. The resulting structure is shown in Figure 1.2.

The SHEL (Software, Hardware, Environment, Liveware) model that was proposed by Edwards (1972) presents a somewhat similar picture shown

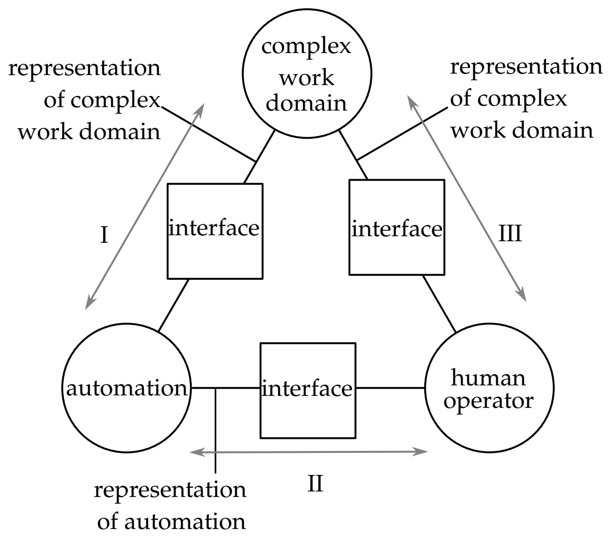

complex work domain
representation of complex work domain
representation of complex work domain
I
interface
interface
III
automation
interface
human operator
representation of automation
II

Figure 1.2: The relationship between the human operator, the work domain, and the automation, and the three interfaces that can be defined in between them.

in Figure 1.3. This model suggests a system view where all the productive processes are mediated by three components.

- The hardware represents physical and non–human components of the system, such as equipment, tools, manuals, signs, etc.
- The software represents components such as rules, procedures, policies, knowledge and practices defining the way in which the different components of the system interact with each other.
- The liveware represents human components in the system in terms of relations and communication with other humans and components.

These components interact in a socio–technical environment that affect the functioning of the system. According to the model, the analysis of the socio–technical systems should focus on the interactions among these components. However, the SHEL model does not address automation explicitly. In the SHEL model, 'automation' would be part of both the 'hardware' (tools) and 'software' (rules, procedures) components. The ecological approach makes the automation component explicit, and its relationship with the complex work domain and the human operator.

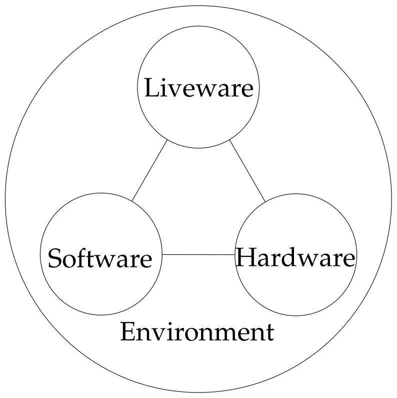


Figure 1.3: The SHEL model shows the interactions of four components of a man-machine system: *liveware* represents the humans, *hardware* represents the physical properties, and *software* represents the non-physical and intangible properties (not to be confused with computer software only). All parts interact with the environment. Adapted from Edwards (1972).

Classical expertise domains

Three interfaces are shown in Figure 1.2. The question "how to represent the complex work domain?" and "how to represent the automation?" can be asked for these interfaces. The questions are usually answered from separate expertise domains.

The first domain of expertise (I) is that of engineers. Based on their understanding of the control problems that need to be solved, automation is designed including the corresponding interface between the work domain and the automation. It is not uncommon that this is a purely technical approach and that only at a later stage in the design, psychologists or "human factors experts" are asked to provide a good human–machine–interface design. The psychologists approach the problem from the second expertise domain (II) and use their understanding of the human operator to design the human–machine–interface. The Cognitive Systems Engineering (CSE)(Rasmussen, Pejtersen, & Goodstein, 1994) and EID paradigms are multi–disciplinary and belong to the third expertise domain (III). An understanding of a technical work domain is combined with an understanding of human information processing capabilities to design better interfaces, in the broadest sense, between the human operator and his work domain. However, the design of automation and the complexity it introduces in the work domain are not addressed by any of these expertise domains.

Shared work domain representation

The next step is to find a structure that allows us to limit the adverse complexity that is introduced by the automation. It is hypothesized that for effective human–automation interaction, the human and the automation should base their reasoning and control actions on the same domain representation as outlined for autopilot design by Amelink, Mulder, van Paassen, Lintern, and Solodilova-Whiteley (2006). This can best be illustrated with the famous ant–on–the–beach parable presented by Simon (1981). By considering the path taken by an ant on a sandy beach, Simon said (p. 64):

> *"Viewed as a geometric figure, the ant's path is irregular, complex, and hard to describe. But its complexity is really a complexity in the surface of the beach, not the complexity in the ant."*

This parable is used by Rasmussen et al. (1994) to build the case for Work Domain Analysis (WDA). WDA is used to make a map of how the work domain constrains [3] the human operator.

However, we use it here to illustrate an underlying message, one that applies to the design of automated systems. Initially the ant's path seemed complex and hard to understand. That apparent complexity disappeared when the observer sees and understands how the beach shapes the ant's path. The ant's behavior can be understood by understanding its goals, and by seeing how it is constrained in pursuing them, but without knowing or understanding the internal workings of the ant. Therefore, complexity is not an objective feature of a work domain but it depends on the available information and how it is represented to the human operators.

We can look at automated systems in a similar way. The behavior of (properly designed and functioning) automation can be understood when the constraints that it works within are made explicit. The internal workings of the system do not need to be understood. Dennett (1981) describes this view as the 'intentional stance' towards systems whose complex behavior cannot be understood based on bottom–up reasoning. This is similar to the

[3]Section 2.2.2 introduces constraints. In short: constraints limit the freedom that one has to achieve goals.

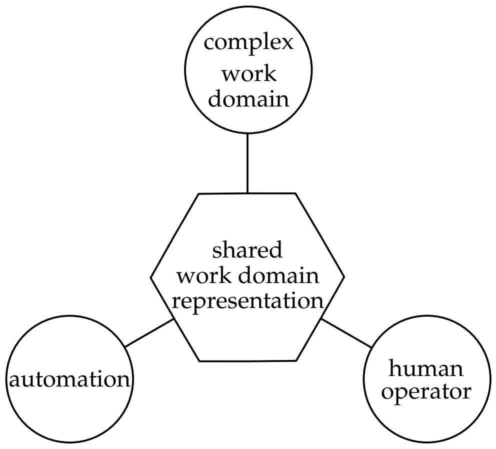


Figure 1.4: The concept of the shared domain representation: the automation and the human operator interact with the work domain through the same representation of the work domain.

suggestion made under the fourth irony of Section 1.2.4: to support monitoring at some meta–level.

We define the concept of the *shared domain representation*. It is used conceptually to visualize and to set the direction of the ecological approach to automation design, in this thesis and for future work. The philosophy of the shared domain representation is that the human operator and the automation work with the same representation of the work domain. The interfaces between the human operator, the complex work domain and the automation are treated as a single entity, as illustrated in Figure 1.4.

The shared domain representation is an explicit representation of the constraints that are imposed on actions. The automation is designed to work within these constraints, and the human operator can understand, predict and verify the automation's behavior based on the constraints. The different internal workings of the automation and the human operator do not present an obstacle in this approach. However, properly functioning automation is at the basis of these assumptions. Thus, issues arising from erroneous design and/or implementation, belonging to the first irony, are not further addressed.

The *law of requisite variety* asserts that "for a system to achieve stability in the midst of perturbations, its number of control variables must be greater than the number of possible states in that system" (Ashby, 1956), placing a minimum on the automation's complexity. Remembering Albert Einstein's quote "Everything should be made as simple as possible, but not simpler" (Wikiquote, 2010), and combining with Ashby's law: the complexity of the automation should at least match the complexity of the work domain and preferably not exceed it.

Three guidelines follow from this.

- The constraints in the work domain are the basis for the shared domain representation.
- The constraints that the automation introduces should be limited or eliminated.
- The constraints that the automation introduces should be made visible and understandable to the human operator.

1.4.3 Work domain analysis prior to design

WDA is used to generate a map of the constraints in the work domain. It is the primary tool used to define the shared domain representation. WDA is the first step of CSE, EID and Cognitive Work Analysis (CWA) (Vicente, 1999). In these approaches, it is typically applied to *existing systems* with the aim to improve the performance of human operators under the current way of working by improving parts of the system and/or the (ecological) interfaces.

In the ecological approach to automation design, our aim is to apply WDA to new, to–be–designed systems intending to limit the automation introduced complexity prior to design. No references have been found to work that describes thoroughly how WDA was applied *prior* to design of a system and automation. The two research questions above are addressed by studying the application of WDA to automation design. This is achieved by developing WDA for automation design through studies into concrete applications, in this thesis primarily vehicle control related.

One of the main challenges to this approach is to keep track of how the automation, that is being designed, is transforming the work domain. The magnitude of this challenge becomes clear when we keep in mind that when automation fails, the transformations that the properly functioning automation introduced are reversed. The human operator, who has to intervene, is presented with a tremendous task. Not only does he have to switch from supervisory control to active control (the third irony), he also has to deal with a system that has now, most likely, changed behavior compared to when he was monitoring it (as illustrated by the introductory quote). By keeping track of the changes to the work domain that automation introduces, the representation can be used to support the human operator to monitor and intervene. We continue the study of the ecological approach taking this challenge into account.

1.5 Outline of the thesis

With the motivation and goals of the research project being discussed in the previous section, the content and structure of the underlying thesis can be described below. First, we briefly discuss which topics this thesis does not contain.

What this thesis does not contain
In the definition of the goals and research questions asked above, the approach to the discussed issues has already been chosen. The approach itself is not the subject of the investigation, and this thesis does not contain a comparison or evaluation of different approaches that could have been taken. The theoretical foundations of the described approach are accepted as axioms. This allows us to focus on the application and development of WDA specifically. Each of the presented case studies deals with a different domain. The intention is to use these domains to generate insight and further develop WDA within the scope of the presented approach. It is not the purpose of this work to present original or new automated solutions, but to present a holistic view of the system's functions as part of the conceptual *shared domain representation*. Experimental evaluation of the developed principles has not been found feasible and is not presented. Instead, a theoretical case is built.

The chapters of the thesis present an exploration of the ecological approach and a search for its application to automation design. Figure 1.5 shows a representation of the structure of the thesis. After first presenting the theoretical basis, four case studies are presented. Each study uses a different work domain, and each study generates insight in the application of WDA.

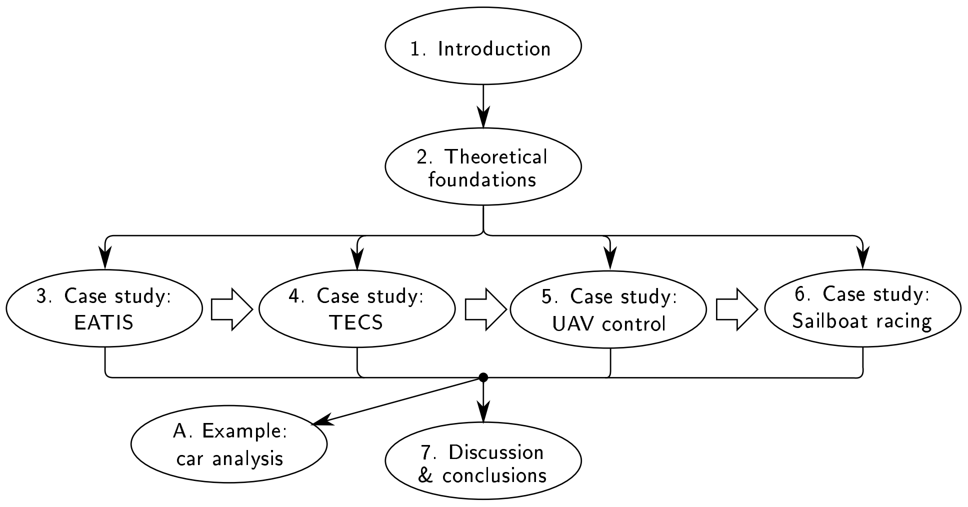

1. Introduction
2. Theoretical foundations
3. Case study: EATIS
4. Case study: TECS
5. Case study: UAV control
6. Case study: Sailboat racing
A. Example: car analysis
7. Discussion & conclusions

Figure 1.5: The structure of this thesis.

The studies are presented in this order to show a progression in the development of the theoretical analysis for automation design. However, as with most analytical processes, there have been many iterations where the analysis of one chapter has led to insights for another chapter. The chapters are structured as follows.

Chapter 2 – Theoretical foundations
The theoretical foundations of the thesis are discussed. The main terminology is introduced and elaborated. WDA is introduced and discussed as the departure point of the exploration and research.

Chapter 3 – Case study: Energy Augmented Tunnel in The Sky display
The Energy–Augmented Tunnel in The Sky (EATIS) display is an example of the application of WDA prior to interface design as part of EID. The work domain is chosen around the manual flight control task. The abstraction hierarchy is used to structure the knowledge obtained by studying the manual flight control task. This led to the identification of the important role of energy management for longitudinal flight control. By following EID principles, this energy management information is represented graphically in the display. A preliminary evaluation of the display provided insights in how pilots experience the new display and generated positive comments.

Chapter 4 – Case study: Analysis of the Total Energy Control System
The design of Total Energy Control System (TECS), conducted in the 1970s and 1980s, is studied as an example of the ecological approach to automation design *avant la lettre*. The approach to the design of TECS has similarities to WDA. The resulting automated flight control system is claimed to have better performance than conventional system at the time (Lambregts, 1983a, 1983b). If those improvements can be attributed to the design approach, it would support the ecological approach to automation design.

The design approach to TECS is well described in literature, and is used to serve as a study to map the different design aspects onto the abstraction hierarchy. This allows the identification of the energy management principles that TECS is based on, and the identification of other, sometimes purely practical, design considerations. A first attempt is made to structure the various control functions in the nested structure of TECS by introducing the levels of control sophistication and the Abstraction–Sophistication Analysis (ASA).

Chapter 5 – Case study: Design of a mini–UAV system
The application of WDA to the new design of a mini–UAV system is studied. The nested structure of the control problems poses challenges to the process of WDA. The lessons learned from the study of TECS are used to adapt the approach to WDA and explicitly support the nested structure through the ASA. This is a hands–on case study that deals with the practical applicability of WDA to automation design of a new system.

Chapter 6 – Case study: The sailboat racing domain
The final case study takes the topic of WDA out of the aviation domain and out of the context of automation design. The developed approach is applied to a highly structured work domain, but one that lacks automation and all actions are taken by a team of human operators. This study shows that the developed approach is not specific to automated work domains but to the analysis of nested control functions.

Chapter 7 – Discussion and conclusions
The four case studies are brought together by recapitulating and discussing the insights generated by them. A comparison is made between the applications and adaptation of the analysis framework, the original framework and similar approaches. The developed analysis framework is discussed in terms of the properties it has, and a generalized approach is formulated based on the case studies. Furthermore, it reflects on the limitations of the

developed framework and applied analyses, and gives recommendations for future investigations.

Appendix A – Example: the car as a transport system
An existing example of WDA with the abstraction hierarchy is expanded with the levels of control sophistication. The original example (Burns & Hajdukiewicz, 2004) shows the car as a transportation system with emphasis on the internal workings of the car. By expanding the analysis space with the levels of control sophistication, the nested structure of the transportation domain becomes visible. The resulting ASA addresses the different control problems ranging from the car as a platform to the transportation system as a whole. This example demonstrates how the ASA provides a much larger analysis space than the original single abstraction hierarchy and how this space can be used to explore the control problems in the domain.

2 Theoretical foundations

The previous chapter discussed the approach to Ecological Automation Design (EAD) and introduced the theoretical basis of EAD. This chapter further details the theoretical basis found in Cognitive Systems Engineering (CSE). First, the background of the ecological approach is discussed. Second, the definition of the concepts and terms that are used in Work Domain Analysis (WDA) are given. Third, WDA is discussed and a number of known issues with WDA is discussed. Finally, some considerations for conducting WDA are presented.

2.1 Background

CSE has its roots in the experiences with safety systems in industrial process plants going back to as early as the 1960s. The analysis of accidents in safety–critical systems, especially nuclear power plants, led to a search for an integrated approach to the design of human–machine systems. The rapid development of new technologies and the growing complexity of systems, required a new approach to the analysis, strategic planning, design, and evaluation of socio–technical systems (Rasmussen 1986).

The current field of CSE is the result of a search for finding models for engineering and design. The foundation of CSE lies in the work of Rasmussen et al. (1983, 1986; 1994). Vicente's (1999) work on Cognitive Work Analysis (CWA) adds to the body of this research. The philosophy of these approaches is that the primary role of workers is to act as flexible and adaptive problem solvers. These approaches are conceptual, meaning that they are not a cookbook for system design or performance improvements, but that they aim to provide a systematic and conceptually coherent basis for uncovering design requirements. CWA focuses on the analysis of the role of the human operators in existing systems, while CSE focuses on the models needed for improving system design for the human operator. Ecological Interface Design (EID) (Vicente & Rasmussen, 1992) is a design paradigm for interfaces and provides more concrete guidelines for the design. CSE, CWA, and EID have a strong focus on the improvement of the 'information systems' that provide the interfaces between the underlying processes and the human operator. The improvement of the core processes in a system is not addressed by these approaches. Additionally, they have a significantly larger focus on the analysis of existing system than on guiding the design of new systems. The common foundation of these approaches is WDA. As explained in Chapter 1, our aim is to further develop WDA to address the *ironies of automation* not just at the interface but at the core processes of modern human–machine systems. The three main models that are used in CSE, CWA, and EID are the *abstraction hierarchy*, the *decision ladder*, and the *Skills, Rules and Knowledge (SRK) taxonomy*.

The abstraction hierarchy is the framework underlying WDA. It is used to identify how the work domain constrains actions of human operators or automated control. By developing an abstraction hierarchy representation for a work domain, a designer or analyst can identify the information that workers need to cope with, over the entire range of operating demands, including unanticipated events (Vicente, 1999). The abstraction hierarchy, being a framework, is transformed into a model of the work domain through analysis. As such, it is presented as an externalized mental model that can be used to support problem solving activities of human operators in their work domain. It is also used to visualize problem solving activities and mental strategies (Rasmussen, 1986).

Rasmussen et al. (1994) (p.173) identify that traditional human factors guidelines are extremely limited during the actual design projects. To give some guidance to the design process, they also present the abstraction hierarchy as a framework to map out the *design territory*, or the work domain of a systems designer. With this model, the designer is aided by having a *navigational outline* of the design problem. Although presented in little detail, it is an indication that the abstraction hierarchy and WDA can serve automation design. The properties of the abstraction hierarchy are further discussed in

Section 2.3.1.

The decision ladder is used to analyze a human operators decision sequences and mental strategies. It structures decision sequences with standardized routines and knowledge states. A decision process can be mapped onto the standardized nodes of the decision ladder. The model supports formal decision making that follows the sequential nodes of the decision ladder. Heuristic decision making is supported as shortcuts between the nodes of the decision ladder. The formal decision making is associated with knowledge based behavior, and the heuristic decision making is associated with rule based behavior. The decision ladder is used for activity analysis (in CSE: Vicente, 1999) and control task analysis (in CWA: Rasmussen, 1986). It has more significance to analysis of work with existing systems than the design of new systems.

The Skills, Rules, Knowledge (SRK) taxonomy is a model for *human* information processing (Rasmussen, 1983). Three levels of cognitive information processing are identified: skill based behavior, rule based behavior, and knowledge based behavior. Skill based behavior represents human sensori–motor performance, which is a highly automated and smooth pattern of behavior without conscious control. Rule based behavior represents the conscious processing of familiar situations with stored rules. Knowledge based behavior represents the processing of unfamiliar situations that require explicit goal formulation and model based reasoning. A high cognitive workload is associated with knowledge based behavior, while a low cognitive workload is associated with skill based behavior. The SRK taxonomy has its place in the analysis of the (social) organization of work in CSE and CWA. As part of EID, it is used to formulate the goal of EID: to not force the cognitive processing to a higher level than needed for the task. The decision ladder supports the representation of the skill based, rule based, and knowledge based behavior in decision processes. The abstraction hierarchy is used as a model that supports knowledge based information processing.

These models are used to match the information processes of computer information systems to the mental decision processes of an operator. This approach does not imply that computers should process information in the same way as humans do. To support human decision making and supervisory control, the results of computer processing must be communicated at appropriate steps of the decision sequence and in a form that is compatible with human decision making strategies (Rasmussen, 1986). The contribution of WDA to these goals is further discussed in Section 2.3

2.2 Terminology

The theoretical basis of WDA has its own particular terminology. Most terms will be introduced during the discussion of WDA, but four terms are introduced in advance. Their meaning forms the basis for defining the content of the analysis and its delimitation. WDA is especially intended for the analysis of complex socio–technical systems. We discuss the *work domain*, *constraints* in the work domain, the *system* and *complexity*.

2.2.1 The work domain

Rasmussen et al. (1994) describe the work domain as "the landscape within which work takes place". This landscape is independent of the human operators, events, tasks, goals, and new automation or new interfaces as was discussed in Chapter 1. The work domain is defined by its boundary. In the original process control domain, the *system boundary* and the *work domain boundary* are often treated as the same. The explanation can be found in the relatively weak interaction between a (power) plant and the direct environment.

However, here we make a distinction between the two boundaries. As Figure 2.1 shows, the system falls within the work domain boundary. The work domain includes the system and the part of the environment that is relevant to the analysis. Therefore, the work domain boundary separates the elements in the environment that are included in the analysis and those that are excluded. It is useful to choose the work domain boundary to have only weak interactions with the elements of the environment around it (Vicente, 1999). The interactions across the system boundary are much stronger than those across the work domain boundary.

In principle, the human operator is not part of WDA, which represents only the *landscape in which work by the human operator takes place*. Similarly, and as argued in Chapter 1, the to–be–designed automation cannot be part of WDA either, because WDA represents the *design territory* for automation design. However, automation is at the basis of the functioning of modern systems. Excluding all automation from WDA would also exclude a lot of functions on which the new and higher level automation will be based. For example, a modern car has computerized engine control to improve its performance and efficiency. The design of an electronic cruise control system for a car will be based on the functional block that the engine and its com-

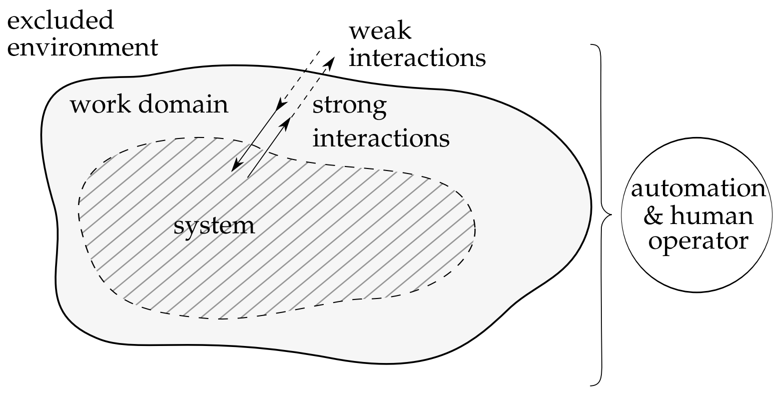


Figure 2.1: Work domain delimitation for WDA. The system is the part of the work domain. It is subject to control by the human operator and/or automation. The automation and human operator are not included in the WDA.

puter form. Therefore, the computerized engine control should be part of the WDA when it is not part of the cruise control design effort.

Therefore, a distinction is made between the *to–be–designed automation* and the *fixed automation* that is not part of the design effort. The latter is regarded as part of the work domain in this approach, while the first is not.

2.2.2 Constraints

WDA aims to describe how behavior of a human operator or automation is constrained by the work domain. In other words: the actions that an operator or automation performs is determined by his goals and the *constraints* that are imposed on his actions. Vicente (1999) uses the following definition:

> Constraints: Relationships between, or limits on behavior. Constraints remove degrees of freedom.

Constraints need to be taken into account when defining behavior for automated systems. For example, *speed protection modes* are implemented in automated flight control systems to ensure that the aircraft does not fly outside its safe speed range.

2.2.3 The system

The system boundary defines the system. The system is that part of the work domain that is being controlled. Usually this is the man–made part of the work domain such as a powerplant or an aircraft. As shown in Figure 2.1, the system has interactions with the rest of the work domain. Different systems interact in different degrees with the rest of the work domain. Vicente (1999) makes the distinction between open and closed systems.

Closed systems are isolated from their environment and have weak interactions across the system boundary. The influences from outside are limited in number and variability, and the system can be well understood by examining its internal functioning. An example of a closed system is a domestic laundry washing machine. We can draw a system boundary around it to include the washing machine only, including its washing programs. Interactions across the system boundary include the supply of laundry, soap, water and electricity. When their supply is guaranteed, and when the laundry's dirtiness is within reasonable limits, and when the machine works as designed, the internal washing programs will deliver clean laundry. The limited and invariable interactions with the work domain allow simple automation with predictable results.

Open systems are subject to influences from outside, and have strong interactions across the system boundary. An example of an open system is an Unmanned Aerial Vehicle (UAV). The work domain around the UAV system includes many dynamic elements. Examples include the air it flies (e.g., wind, temperature, air pressure), the world it navigates in (e.g., obstacles, navigation aids, launch and landing sites), the airspace it shares with other aircraft (e.g., collision avoidance), and the mission it needs to fulfill (e.g., imaging of a moving target). The difference with closed systems is that these elements cannot be directly controlled and the controlled system has to react to them. The large number of interactions across the system boundary and their variable nature pose great challenges for automation design. It is the main reason why passenger aircraft are not fully autonomous yet.

2.2.4 Complexity

The word *complexity* has been used as a feature of a work domain that poses challenges for control and automation design. WDA has been proposed to describe work domain complexity to allow us to limit automation added complexity. But what is complexity? A notion of complexity is given to be able to work with the application of the framework of WDA to complex work domains. It is not our goal to adopt or provide a definition or model of complexity, neither is this notion complete. If a definition or model was used, the focus would inadvertently shift towards theoretical descriptions defeating our goal [1].

The main contributing factors to complexity are identified by reviewing descriptions of complexity in evolution in nature by Edmonds (1999). This source is unbiased by automation design issues, and stays close to the ecological approach. The contributing factors are presented as independent from their sources. Vicente (1999) gives an overview of sources of complexity in systems. The contributing factors are discussed from the WDA perspective.

Size and interrelations – Many elements in a work domain do not make the work domain complex, but a certain number of elements is needed for complexity. The interrelations between the elements introduces complexity. For example, one hundred aircraft flying in random directions in a shared airspace pose more complexity to the air traffic controllers than one hundred aircraft flying in the same direction. Not only the number of aircraft but also the degree in which aircraft constrain the flight of other aircraft contributes to complexity.

Variety – Non-uniformity of the elements found in the work domain, thus the number of different types of elements in the work domain imposes complexity. Different elements impose different constraints on actions. For example, the departure of a light general aviation aircraft amidst departing commercial airline jets requires different treatment due to its slower speed and other factors such as the higher sensitivity to wake vortices. The different constraints that the general aviation aircraft imposes on the work of the air traffic controllers increase the complexity of their work.

Disorder – An ordered work domain, thus one with structure, poses less complexity than a chaotic work domain. In a structured work domain, the pattern of the structure can be used to describe the information in the work domain in less space than an equally large but less structured work domain. This is closely related to the minimal description size, which is the minimum

[1]Like 'complexity', 'quality' is an abstract concept. Pirsig (1974) writes about the metaphysics of quality explicitly without defining 'quality' in concrete terms because he argues that 'quality' will then loose its meaning.

possible length of a description in some language, also called Kolmogorov complexity (Edmonds, 1999). When the information (in a work domain) can be compressed, the complexity is smaller than an equally large description that cannot be compressed, or compressed in a lesser degree.

Ignorance – The lack of knowledge of the processes that run inside a system and work domain can increase apparent complexity. For example, a cockpit will appear more complex to the student pilot than to the experienced pilot. The latter one has built an understanding of the processes that interrelate the various elements in his work domain.

The contribution of *ignorance* to complexity shows that complexity is not an objective feature of a system or work domain. Complexity is a perceived property of the work domain and therefore depends on the observer (Rasmussen, 1986). Objective complexity can only be defined for a given representation of a work domain, not for the work domain itself (Rasmussen & Lind, 1981). The apparent complexity is especially relevant to the ecological approach because the inherent properties of the work domain cannot be changed (and should not be simplified) but its representation can be chosen, which is an opportunity to reduce the apparent complexity. This list is appended with two additional subjective contributing factors.

Lack of representation and information processing – The ability to perceive and process information in the work domain decreases apparent complexity. In Chapter 1 this contributing factor was illustrated using the ant–on–the–beach parable. Once the constraints that the beach imposes on the ant's behavior were identified, the seemingly complex path of the ant was understood and the apparent complexity was reduced. Therefore, the apparent complexity depends on the information that is available, the observer's ability to perceive and to process the information.

Intentions and goals – Pursuing intentions and goals in a work domain makes it more complex. Intentions and goals are associated with the purpose of a system and the task of an operator. Without goals, one would not be constrained by the work domain. For example, the air traffic controllers dealing with one hundred aircraft in the same airspace find the work domain complex because their goal is to keep the aircraft separated. Without this intention the interactions between the aircraft would not be of interest and the complexity would disappear.

2.3 Work domain analysis

If the work domain is the *landscape within which work takes place*, then WDA is the process of creating the *map* of that landscape. The aim of WDA is to produce a "generalized representation of the work domain in terms of its inventory of objectives, functions, activities, and resources – all of which constitute the elements of the landscape in which the staff operates" (Rasmussen et al., 1994). This map is a representation of survey knowledge rather than route knowledge. It is a map of how the work domain constrains actions that an actor can take to reach goals similar to how a street map shows how the existing roads constrain a driver's actions to reach a destination. WDA is a study of the work domain that is independent of the workers or the automation that control processes in the work domain. The abstraction hierarchy is fundamental to this approach.

2.3.1 The abstraction hierarchy

The abstraction hierarchy is a set of levels, each with a different representation of the work domain. The levels range from concrete to abstract and are connected by the means–ends relationship. It allows us to identify functions at different levels of abstraction and their interrelations in a work domain.

The type of hierarchy

The abstraction hierarchy belongs to the class of stratified hierarchies described by Mesarovic, Macko, and Takahara (1970). The properties of the stratified hierarchy are listed below:

1. Each stratum, or level, of the hierarchy deals with the very same system, the only difference being that different strata provide different descriptions, or different models for observing the system.
2. Each stratum has its own unique set of terms, concepts, and principles.
3. The selection of strata for describing a particular system depends on the observer, and his knowledge and interest in the control of the system. For many systems, however, there may be some strata that appear to be natural or inherent.
4. The requirements for proper system functioning at any level appear as constraints on the meaningful operation of the lower levels, while

the evolution of the state of the system is specified by the effect of the lower levels on the higher levels.

5. Understanding of the system increases by crossing levels: by moving up the hierarchy, one obtains a deeper understanding of system significance with regard to the goals that are to be achieved, while moving down the hierarchy, one obtains a more detailed explanation of the system's functioning in terms of how those goals can be carried out.

The relationship between the levels

In addition to the characteristics of the stratified hierarchy, the structure of the abstraction hierarchy is further specified by a means–end relation between the levels. Its this explicit goal–oriented nature of the abstraction hierarchy that sets it apart from other hierarchies and adds the important psychological implications (Rasmussen et al., 1994; Vicente & Rasmussen, 1992). The functions at one level of abstraction are the means to achieve the functions at the next higher level of abstraction. The reason *why* one function is defined at one level is described by the level above it, and the way *how* it is achieved is described by the level below it. Rasmussen (1986) describes this with the *why–what–how* relationship between the levels, as illustrated in Figure 2.2.

The means–ends structure allows the structuring of a technological system in terms of how functions are implemented to achieve higher level functions, or goals. This happens in every design but not necessarily systematic. The use of the abstraction hierarchy encourages explicit attention to the higher levels of abstraction.

The levels of abstraction

The top level represents the system's design purpose in its environment. The physical implementation is represented at the bottom level. In between, the levels represent the system with intermediate degrees of abstraction.

The number of levels in the abstraction hierarchy as well as their content will vary from domain to domain depending on the work domain and the interest of the analyst. Although there are differences between the vehicle control and the process control domain, Rasmussen's five levels are adopted throughout this thesis. They are well described by Rasmussen and have proven to be a good starting point for the analysis in the aviation domain

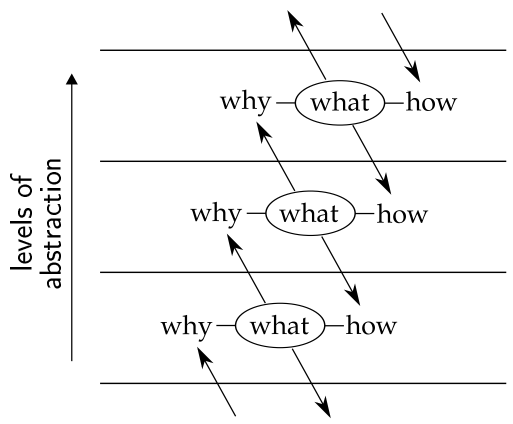


Figure 2.2: The means-ends relationship between the levels of the abstraction hierarchy is characterized by *why–what–how* (based on Rasmussen (1986)).

(Amelink, Mulder, van Paassen, & Flach, 2005b; Borst, Suijkerbuijk, Mulder, & van Paassen, 2006; van Dam, Mulder, & van Paassen, 2004; Flach, Patrick, Amelink, van Paassen, & Mulder, 2003; Dinadis & Vicente, 1999).

The main difference between the process control domain and the vehicle control domain is the distinction between closed and open system as discussed in Section 2.2.3. A process control system, such as a power plant, is a fairly closed system and the complexity primarily originates from within the system. The system boundary and the work domain delimitation virtually coincide. The terms 'system' and 'work domain' are almost used synonymously throughout CSE and CWA literature. In contrast, a main component of the complexity in the vehicle control domain comes from the many and unpredictable interactions between the system and the environment.

Below the five levels are summarized taking into account the original usage of terms. The levels and their general content are discussed according to Rasmussen (1986) and Rasmussen et al. (1994).

Functional purpose level – This level is also referred to as the 'system purpose' level or the 'purposes and constraints' level (Rasmussen et al., 1994; Rasmussen, 1986). This is the highest level of abstraction. It describes the purpose of the system in relation to its environment. This can be in terms of simple quantitative input–output relationships or the functional relationship between system and environment described by policies and strategies (Rasmussen et al., 1994). This level defines the interactions across the system boundary.

Abstract function – The overall functioning of a system is represented by a generalized causal network, e.g., in terms of information, energy, mass, or monetary flow structures reflecting the intended operational state. The laws and organizing principles at this level form a consistent structure independent of the system that is satisfied by its design. These can be regarded as a *reason* from which system properties can be derived (Rasmussen, 1986). For example, the laws of conservation of mass and energy can be used for understanding system behavior at lower levels of abstraction. The descriptions of the elements reflect their mutual dependency by co–functioning of all elements.

Generalized functions level – The work domain is represented in terms that are familiar for the particular domain. The concepts and functional relationships are independent of the underlying physical implementation. The descriptions reflect the properties that allow the co–functioning of components at the lower, physical function level. In process control this could include terms as: 'power conversion', 'heat exchange', 'safety system'. There is normally not a one–to–one relationship between the physical function of a component and its generalized function. Typically there are many–to–many mappings (Rasmussen, 1986).

Physical function level – At this level the function of components is described in terms of its physical processes. The representation focuses on specific physical equipment and physical variables used to characterize functional states. It describes the functions that are inherent to the physical process and its limitations. At this level, the causes of malfunction are typically identified. Physical changes, like malfunctions, propagate up through the levels of abstraction (Rasmussen, 1986).

Physical form level – This is the lowest and most concrete level of the abstraction hierarchy. The representations at this level describe the physical appearance, material and configuration of the system. This can include blueprints, maps of physical layouts, diagrams, pictures, etc. These descriptions are vital to identify parts and components in a system, e.g., when trouble shooting a system (Rasmussen, 1986).

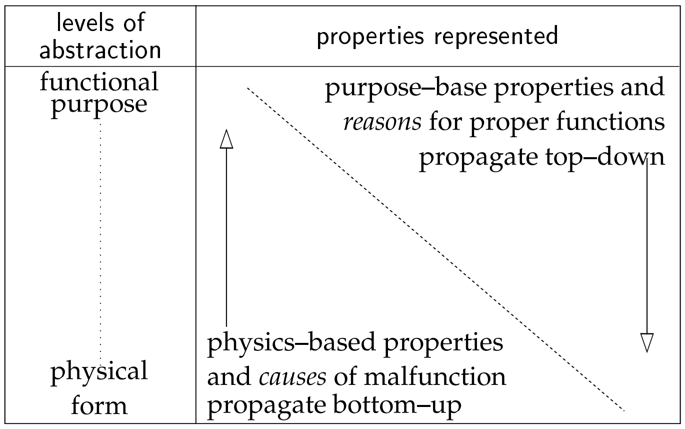

levels of abstraction
properties represented
functional purpose
physical form
purpose–base properties and *reasons* for proper functions propagate top–down
physics–based properties and *causes* of malfunction propagate bottom–up

Figure 2.3: Levels of abstraction in system representation, adopted from Rasmussen et al. (1986).

The overall structure of the means–ends space is illustrated in Figure 2.3. It shows that the functional and material features of the work domain dominate the representation at the lower levels, and that the objectives that govern control of the system dominate the higher levels (Rasmussen, 1986). It also shows that changes in the objectives propagate down through the levels, making control concrete. Changes in the physical basis of the work domain propagate up, potentially interfering with the objectives.

2.3.2 The abstraction–decomposition space

The Abstraction–Decomposition Space (ADS) is created when a second dimension is added to the abstraction hierarchy. Along this dimension, the whole–part decomposition of the system into subsystems takes place. The result is a two–dimensional space with levels of abstraction along the vertical axis and part–whole decomposition along the horizontal axis.

The relation between abstraction dimension and decomposition dimension is very complex. The decomposition at one level might not be possible at another level. The ADS is described by Rasmussen et al. (1994) as non–linear, meaning that a step in level of abstraction followed by a step in decomposition will not give the same result when the same steps are taken but in reversed order.

The most common representation of the ADS is a two–dimensional matrix with the levels of part–whole decomposition cutting across the levels of abstraction. Examples are the representation of the DURESS II micro world in Vicente (1999), the fuel and engines system of a Hercules aircraft in Dinadis and Vicente (1999), the computer repair example in Rasmussen et al. (1994), and a UAV mission domain in Castro and Pritchett (2005). The ADS is defined by Vicente (1999) (p.158) as: each cell in the two–dimensional space is a place holder for a different but *complete* representation of the same work domain. Often this approach results in a detailed diagonal in the two–dimensional matrix with empty corners, which is essentially a one–dimensional representation of the work domain. This has also been addressed as an issue of the ADS by Lind (2003).

According to Rasmussen's (1986) original view on the ADS, the structural decomposition should take place separately for the levels of abstraction. Only when the analysis is made for a *specific situation*, such as Rasmussen's computer repair example, the same part–whole decomposition can be applied to each level of abstraction. In the following chapters, we will apply the part–whole decomposition per level of abstraction. In the following sections, the term 'ADS' is only used to denote the two–dimensional matrix where levels of decomposition cut across all levels of abstraction. The usage of the term 'abstraction hierarchy' implies part–whole decomposition taking place per level of abstraction.

2.3.3 Psychological relevance

Arguments for the psychological relevance of the abstraction hierarchy for problem solving is presented by Vicente and Rasmussen (1992) as part of EID, and Rasmussen et al. (1994) as part of CSE. The evidence is not repeated, but the underlying structure is discussed.

Unanticipated events

Unanticipated events are, by definition, problematic situations that were not foreseen by designers. Therefore, systems are not designed to deal with their occurrence and the individual human operator, who happens to be supervising the system when the event occurs, has to engage in solving the problem. The operator needs to be able to reason about the system's functioning. The occurrence of these events and their potential consequences form a big threat to safety and are addressed in the CSE, WDA and EID approaches.

How can a system give support to the human operator for dealing with situations that were not foreseen by the designers? The answer is that the operator can be supported with a generalized representation of the work domain that supports reasoning but that is not specific to any task or situation. The content of the abstraction hierarchy in terms of constraints, and the lack of specific tasks, actions and procedures provides a generalized description. Due to its means–ends structure, the abstraction hierarchy provides a basis for goal directed reasoning and problem solving that is needed for coping with unanticipated events (Vicente & Rasmussen, 1992). This is in contrast to decision trees that address each possible failure in the systems and specifies actions to be taken in each situation, leaving the unanticipated events unaccounted for, by definition.

Reasoning with the abstraction hierarchy

Causal, bottom–up reasoning is not possible for all the environments in which humans have to make decisions. Systems with a high degree of autonomous internal functioning, with self–organizing and highly adaptive features, will change their internal functional organization continuously in order to meet the requirements of the environment and to suit their internal goals or performance criteria (Rasmussen, 1986)(p. 117). Decision making in control of complex systems is based on the knowledge of the organizing principles of the system, its coupling to the environment and its limitations. Thus, it is based on reasoning top–down in the abstraction dimension, with little or no consideration of the internal causal structures or functions (Rasmussen, 1986).

As discussed in Section 1.4.2, when system complexity rises such that the human mind is no longer able to effectively reason cause–effect or bottom–up, human reasoning for predicting behavior must be top–down. By taking the *intentional stance* rather than the *physical stance* or the *designer stance*, the behavior of the system can be understood without reasoning explicitly about the internal processes of the system (Dennett, 1981). Thus, the behavior of intentional systems are understood based on their intentions (or goals, or purpose) and the constraints within which it pursues its intentions. The concept of *intentional model* is used by Dennett (1971) to connect the intentional domains (including common sense reasoning of human operators about a system) and the non–intentional domain of physical sciences (including the design of systems). This approach is very similar to viewing a system on different levels of abstraction and in terms of the constraints on system behavior.

The abstraction hierarchy connects the elements of the work domain in a means–ends manner so that they can be seen in relation to what their meaning or purpose is. The problem solving itself is constrained to that which is relevant by starting on a high level of abstraction, moving down only concentrating on the subset of the domain that is connected to the function of interest. Shifting in level of abstraction at which a control problem is viewed, can be very effective in solving a problem situation (Rasmussen, 1986). Rasmussen (1986) has found that "formal reasoning appears to be horizontal reasoning within a single level of the hierarchy" and that "practical functional reasoning is related to vertical transformations in the abstraction hierarchy".

Relevance to design

The goal–oriented means–end structure of the abstraction hierarchy is also relevant to the design of systems. Rasmussen (1986) argues that "system design is a process of iteration between considerations at the various levels rather than an orderly transformation from a description of purpose to a description in terms of physical form". Similar to how the troubleshooting trajectory of the maintenance engineer was mapped onto the ADS, the design trajectory during a design process can be mapped onto an ADS (Rasmussen et al., 1994). Furthermore, the abstraction hierarchy can be used as a way to organize the knowledge about the work domain prior to design, and as a means to "visualize the knowledge base to a user by displays made available for a particular activity" (Rasmussen et al., 1994). However, their discussion focussed on the design of the information system to control the underlying processes, which is very similar to the design of (ecological) interfaces. Our aim is to focus on the design of the underlying automated processes.

When an automated system is framed as an information–processing device, Marr's (1982) levels give support for understanding the automated process. Marr presents three levels at which an information–processing device must be understood before one can be said to have understood it completely. "For far too long, a heuristic program for carrying out some task was held to be a theory of that task, and the distinction between what a program did and how it did it was not taken seriously." The three levels at which a machine's information–processing task must be understood are (Marr, 1982):

Computational theory – What is the goal of the computation, why is it appropriate, and what is the logic of the strategy by which it can be carried out?

Representation and algorithm – How can this computational theory be implemented? What is the representation for the input and output, and what is the algorithm for transformation?

Hardware implementation – How can the representation and algorithm be realized physically?

Like the abstraction hierarchy, these levels range from the formulation of the purpose of the system to the physical implementation. The middle level has a strong resemblance to the abstract function level, where the organizing principles and value structures are represented. Although Marr (1982) does not explicitly state it, the levels can be linked through the "why–what–how" relationship that Rasmussen (1986) describes for the levels of the abstraction hierarchy. The compatibility of Marr's levels with those of the abstraction hierarchy lead us to find additional support for the relevance of WDA for automation design.

2.4 Known issues with the abstraction hierarchy

WDA has the potential to represent complexity in the work domain. However, there are some unclarities and potential problems with the abstraction hierarchy as a modeling tool.
Lind (2003) gives an overview of the main methodological and conceptual problems with the abstraction hierarchy that he found. The discussed issues in this section indicate that the application of WDA with the abstraction hierarchy still lacks a well described approach that guides analysis. Below we briefly discuss the main issues.

2.4.1 Methodological problems

The methodological problems refer to the issues with the process of performing the analysis for a particular domain. These are encountered, especially, when starting with the analysis and when the analysis is used as a model of the domain.

Knowledge acquisition

The first point is where to find the knowledge to represent in the abstraction hierarchy (Lind, 2003). This is especially an issue for complex socio–technical system, where knowledge is typically based on the process of interviewing operators and system experts and interpretation of the systems design documents. The process of capturing the required knowledge and then the formulation of the purposes, goals and objectives is unguided by the abstraction hierarchy.

Model building and validation

The second point goes hand in hand with the first point. The formulation of WDA with the abstraction hierarchy lacks descriptions for identifying which pieces of knowledge go on which level of the abstraction hierarchy. This is left to the analyst, who is collecting the knowledge. Furthermore, validating and revising the model is not described either and is, so far, only defined by incomplete examples for a few specific work domains. Basing methodology on these examples is a weak approach because it will inherit the ambiguities and hidden assumptions of the examples.

Control systems

As described in Chapter 1 the representation of automation (control systems) in the abstraction hierarchy is a controversial issue. Lind (2003) writes that this is not a conceptual problem with the abstraction hierarchy but a methodological one that needs to be addressed. This is mainly due to Vicente's (1999) interpretations of the abstraction hierarchy (and the ADS) that are not consistent with Rasmussen (1986). These ambiguities have been discussed in Section 2.3.2. Lind (2003) writes that the proper representation of industrial processes requires consideration of (at least) two types of means and ends that can be represented in two closely coupled hierarchies: one for the process (work domain) and one for control hierarchy. These hierarchies can only be described by a proper identification of the boundary between the process and the control system. This issue will be encountered and addressed in the analysis in the case study of Chapter 5.

2.4.2 Conceptual problems

The conceptual problems with the abstraction hierarchy refer to the concepts used to model. Heterogeneity and generality are at the roots of these problems. The abstraction hierarchy is open to interpretation allowing the representation of diverse concepts and generalization across work domains. However, Lind (2003) identifies this as a serious weakness: "the unclear semantics disguise the real nature of the modeling problem".

The levels of abstraction

The definitions of what is represented at the levels of the abstraction hierarchy, as discussed above, invite a confusion of the levels. Lind (2003) finds that mass and energy representations at the abstract function level are not sufficient for defining overall purpose of powerplants, and that other terms need to be introduced at the abstract function level. WDA for process control (and other domains, see the example in Appendix A) has moved in this direction based on limited and incomplete examples. Rasmussen (1986) does not prescribe that this is the only terminology that belongs to the abstract function level, neither does he prescribe the number of levels of abstraction to be used or their exact labels. The levels are open and *need* to be interpreted by the analyst and designer. However, this informal and open approach leads to the methodological issues that were discussed in Section 2.4.1.
This issue is illustrated by a number of examples by Lind (2003), whether they are issues with the abstraction hierarchy or issues with interpretation and application is not clear. From them we can conclude that the abstraction hierarchy misses ontological clarity: what exactly is represented? It is one of the main obstacles of the application of, and further development of WDA for EAD. In the following chapters we will work towards an unambiguous definition of levels of abstraction that works well for the presented case studies.

The concept of function

Lind (2003) makes the distinction between functions that are ascribed by convention (like the monetary value ascribed to a coin) and the function by disposition (like a pump's ability to pump water by design). The first function deals with the reason for design, while the second one deals with the behavior as a *result* of design. According to Lind (2003), the abstraction hierarchy does not inherently allow this distinction to be made.

Circular means–ends structures

Lind (2003) finds that the abstraction hierarchy does not support the representation of circular means–ends dependencies. The analyst is not guided by the abstraction hierarchy framework to avoid circular means–ends structures. This may, in fact, be the source of confusion when representing control system (automation) in the abstraction hierarchy. In our experience, as is expressed in the following case studies, there is a natural tendency to model feedback and recursive dependencies across levels of abstraction. This would lead to circular means–ends structures and should be avoided as it is not supported by the hierarchical structure of the abstraction hierarchy.

Means–ends and part–whole levels

Finally, Lind (2003) addresses the issue of the diagonal in the ADS. As our discussion above points out, the levels of part–whole decomposition only cut across all levels of abstraction in special cases, as was argued by Rasmussen (1994; 1986). Therefore, this issue is well noted by Lind (2003) but it is really more an issue with the popular examples than the original definition of the ADS; using part–whole decomposition per level of abstraction.

2.5 Considerations for conducting WDA

To finalize the theoretical foundation of the thesis, guidelines for conduction WDA are presented. A selection of Viciente's (1999) (p. 171) list of practical hints is repeated here to illustrate the considerations when using the abstraction hierarchy as a modeling tool.

1. Define the scope of the analysis by defining the work domain boundary.
2. Choose smallest parts of the analysis, thus the highest resolution that will be used in the representation.
3. Along the abstraction dimension start with the top and bottom level, then bridge the intermediate levels.
4. Do not confuse part–whole decomposition and abstraction.

5. Do not confuse action means–ends and structural means–ends, the abstraction hierarchy should only contain the latter, since it may not include actions.
6. When a system has multiple purposes, develop abstraction hierarchies separately and combine them later.
7. Make sure that at different levels use different modeling languages, and at the same level of abstraction the same modeling language is used.
8. Verify the means–ends links through the why–what–how relationship
9. Make sure that the only types of links used are the part–whole and means–ends links.
10. It is an iterative process thus many revisions are needed.
11. Refer to examples found in literature.

As a final note of this chapter, we repeat that the above theory has originated in process control. Its formulation was mainly intended for the design of information systems and interfaces to monitor and control the process plants to improve the human operator's part the safety of these systems. The focus has been on enabling the human operator to be an effective problem solver, and on reducing the apparent complexity of the work domain through representation at the interface. In the following chapters, an investigation is presented into applying this theory to the design of the automated processes with the aim to limit the automation induced complexity.

3

Case study: Energy Augmented Tunnel In the Sky display

This chapter presents the analysis of the manual longitudinal flight control task and the design of a novel display that makes the energy coupling of an aircraft explicit for manual control. The design follows the principles of Ecological Interface Design (EID) with Work Domain Analysis (WDA) at its basis. It serves as an example of the use of WDA as part of the design objective of a novel display for flight control. The analysis includes the properties of flight and does not involve automation design.

Paper title:	Theoretical Foundations for a Total Energy–Based Perspective Flight–Path Display
Authors:	M.H.J. Amelink, M.Mulder, M.M. van Paassen, and J.M. Flach
Published in:	The International Journal of Aviation Psychology, Vol. 15, No.3, 2005, pp. 205–231.

Abstract
One of the most difficult aspects of manually controlled flight is the coupling between the control over the aircraft speed and altitude. These states cannot be changed independent of each other through the aircraft control devices, the elevator and the throttle. Rather, to effectively change an aircraft's speed and altitude, the controls have to be coordinated. The mediating mechanism that underlies the coordination of the controls is the management of the aircraft's energy state. This article shows that the abstraction hierarchy (AH; Rasmussen, 1986) framework can be effectively used to gain more insight into the underlying structure of the aircraft energy management problem. The derived AH representation is based on the analysis of the energy constraints on the control task. It reveals the levels of abstraction necessary to link the aircraft's physical controls to the speed and altitude goals and also how the aircraft energy is a critical mediating state of the control problem. Energy awareness can be increased by presenting explicit energy management information. The powerful and novel concepts of the total energy reference profile and energy angle are introduced in this article and applied in the context of a perspective flight–path display. The resulting display presents energy management information fully integrated with the tunnel–in–the–sky display and reveals 5 new and important energy cues, intuitively linking the controls and the goals.

3.1 Introduction

In his classic textbook on piloting, *Stick and Rudder*, (Langewiesche, 1944) used a concept called *lift*. This is not the lift force generated by the wings as analyzed in aerodynamics and aeronautical engineering. Rather, Langewiesche used it to illustrate an aircraft's potential to fly. He wrote that an aircraft with lots of lift is safe because the aircraft can easily gain altitude or pick up speed, whereas an aircraft with a lack of lift is very limited in maneuvering. The concepts taught in Langewiesche's book today still help student and accomplished pilots better understand the airplane. This article, like Langewiesche's book, strives to provide a deeper understanding of the airplane and the task of flying, but in a cognitive systems engineering context. The key concept of lift that Langewiesche discussed is here identified as total energy. Total energy is the sum of the aircraft's kinetic energy, which is the energy of the aircraft's speed, and the potential energy, the energy in the aircraft's height. We show that understanding the energy management in flight is essential to a deep understanding of flight control.

With few exceptions,[1] today's modern cockpit does not support the concepts of total, potential, and kinetic energy, although they could be of use in learning and performing the task of flying. The introduction of energy–related information in future guidance displays, such as the tunnel–in–the–sky display, has been reported by (Theunissen & Rademaker, 2000), who adopted a basic, symbolic presentation that is common practice in head–up displays (HUDs; Newman, 1995). Similarly, Sachs and Sennes (2001) reported a control–theoretical analysis of further augmenting the HUD–like symbology for energy management.

In this article, a more fundamental and novel approach is chosen. We explore Rasmussen's (1986) abstraction hierarchy (AH) as a framework for modeling the task of flying. Other descriptions of the piloting task, often based on dynamic control–theoretic models, only describe the machine's behavior, not the machine's functions and their relation to the goals to be achieved. In contrast to this, the representation of the energy management task in the AH provides the link among the task objectives; the management of kinetic, potential, and total energy; and the control possibilities (Amelink, Mulder, van Paassen, & Flach, 2003a). Our understanding of the flight task, mapped in the AH, is then used as the avenue for communicating these concepts to pilots, by designing a display that shows the energy relations and their relevance to the flight goals, using the paradigm of ecological interface design (EID; Vicente & Rasmussen, 1992). By mapping the domain constraints to the pilot interface, the AH can become an externalized mental model to enhance pilot energy awareness and energy management (Amelink, van Paassen, Mulder, & Flach, 2003b).

The article consists of two parts. In the first part, the mapping of the flight task to the AH is described. The second part of the article discusses the energy–augmented perspective flight–path display and the translation of the AH mapping to this display.

[1]Some sailplanes are equipped with a total energy–based climb indicator, which shows total energy gains and losses as a climb or descent speed; that is, in the units of potential energy rate. A demonstration version of the energy display can be downloaded from http://www.amelink.net/mscthesis.

3.2 Scope of the workspace analysis

Flach et al. (2003) described a first attempt to capture the task of landing in the AH framework, and for our analysis, this article is used as a starting point to further elaborate the role of the energy constraints. The analysis in Flach et al. took a broad view of the flying task, illustrated by the fact that safe flight was identified as the top goal. A more restricted scope is used in this article.

The top level of the AH, the functional purpose level, defines the system's goal in the environment. A precise definition of the system boundaries will lead to a better insight into the system that is being analyzed. Our interest is the role of energy during the precision landing task and the system goals and boundaries should reflect that. There are basically two goals that the pilot has during (symmetric) flight: following a certain speed profile and following a certain altitude profile. Of course, other tasks, like managing the fuel systems, are needed for the aircraft to function as a whole, but these are left out of consideration in this article. Thus, the functional purpose level is defined by speed and altitude profiles that need to be flown.

On the other end of the AH we find the aircraft–dependent levels, the physical form level and the physical function level, that deal with the physical implementation of the aircraft system. On these levels are, among other things, the manipulators the pilot has for the control of symmetric flight: the throttle and elevator. The coordination of these controls to achieve the speed and altitude goals was one of the main points of interest for Langewiesche: The student pilot has a throttle and an elevator to control speed and height, but which manipulator controls what? The answer is that neither one controls the aircraft speed or altitude independently from the other. Rather, they must be used in coordination. We believe that the key to the coordination of the manipulators lies in controlling the energy state of the aircraft, and this is what should be on the middle levels of the AH. These levels link the means (throttle, stick or elevator) and the ends (target speeds and altitudes) using energy relations. An analysis that makes these relations explicit can be used as a basis for an EID (Vicente, 2002). This approach to flight interface design can be considered a complement to the development of automatic flight control systems by (Lambregts, 1983a), who also recognized the role of energy in flight control.

3.3 Control task analysis

3.3.1 Observed pilot control strategies

During the approach the pilot manages the aircraft state to comply with the speed and altitude goals set by a predetermined trajectory. Atmospheric disturbances interfere with accomplishing these goals and force the pilot to take corrective actions. Depending on the type of aircraft and landing situation, the pilot generally applies one of two control strategies. In the first control strategy, the pilot uses the throttle to control the vertical flight path (altitude) and the elevator to regulate speed. This strategy is referred to as *throttle–to–path and elevator–to–speed*. In the second control strategy, the pilot uses the elevator to control the vertical flight path and the throttle to control speed. This strategy is referred to as *throttle–to–speed and elevator–to–path*. These reflect two different coordinative structures, in terms of which degrees of freedom are locked out and which degrees of freedom are controlled (Bernstein, 1967).

In principle, either strategy might be used to land. However, from discussing the issue with pilots, there appear to be clear preferences. The elevator–to–speed mode seems to be preferred for standard runway approaches (e.g., commercial aviation), whereas the throttle–to–speed mode seems to be preferred for approaches to shortened runways (e.g., aircraft carriers, general aviation). The focus of this analysis is on a landing to a standard precision runway, which is typical for commercial and general aviation. The analysis assumes zero–wind conditions. However, we suspect that specification of the energy state may be particularly valuable in variable wind conditions.

3.3.2 Means for controlling energy

Speed and altitude are directly related to the total and potential energy of the aircraft. To understand the aircraft energy control one must first understand what the energy relations are and how energy can be regulated. *Kinetic energy* is the energy of a moving object and is a function of its speed, as shown by:

$$E_{kin} = \frac{1}{2}mV^2 , \tag{3.1}$$

where m is the aircraft mass and V the aircraft's velocity relative to the ground. The aircraft's potential energy is determined by its altitude above a ground reference such as the runway threshold, as shown by:

$$E_{pot} = mgh \,, \tag{3.2}$$

where h is the altitude above the reference and g is the gravitational acceleration. The sum of the two energies is the aircraft's *total energy E*. The law of conservation of energy states that energy cannot be created or destroyed. This means that when the total energy is constant, the kinetic and potential energies can change but only in equal and opposite amounts. Thus, altitude can be traded for speed and vice versa without gaining or losing total energy. Langewiesche (1944) made this point when he wrote that speed = height and called it the law of the roller coaster. The other implication of the law of conservation of energy is that an aircraft can only lose total energy through drag: The energy is transformed into heat, which is bled off to the surrounding air. The only way in which an aircraft can gain total energy is through the energy added by the engine. The net total energy flow into the aircraft is a function of the difference between engine thrust, T, and the aircraft drag, D:

$$\frac{\dot{E}}{V} = T - D \,, \tag{3.3}$$

where $\dot{E}$ is the total energy rate. Except for the throttle, the pilot only has controls to increase drag, which are of course used as little as possible. Therefore, the engine thrust is the preferred way to control total energy, and since the throttle controls the engine, the throttle becomes the aircraft's energy control. Thus, the throttle does not control speed or altitude like the two control strategies that have been introduced above imply, but rather controls the aircraft's total energy rate.

What does the elevator do? It is reasonable to assume that the elevator has negligible influence on drag, as changes introduced by the flight control surfaces (e.g., elevator) are small relative to the total drag. Furthermore, when the maneuver rates are small (as with commercial aircraft), the variations of induced drag can be neglected as well. Thus, control of the elevator has negligible impact on total energy or total energy rate. What it does do is exchange energy between kinetic and potential energy: It is the energy distribution device. This is where Langewiesche's law of the roller coaster comes into play: When using only the elevator to go up it will be at the expense of speed; in a complementary fashion, speed can be gained at the expense of height.

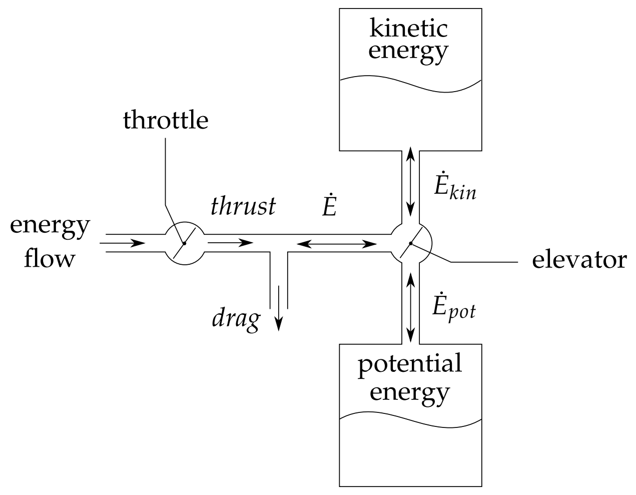


Figure 3.1: The reservoir analogy. The throttle regulates the total energy flow and the elevator controls the energy flow distribution. In this figure, $\dot{E}$, $\dot{E}_{pot}$ and $\dot{E}_{kin}$ represent the total, potential, and kinetic energy rates, respectively.

3.3.3 The reservoir analogy

The energy controls can be visualized as if the aircraft is a system holding two reservoirs. One contains the kinetic energy and the other the potential energy. Together these reservoirs represent the total energy. There is one energy flow into the system provided by the engine and there is only one energy flow out of the system through aerodynamic drag. The net energy flow results from the difference between thrust and drag (Equation 3.3). This flow is then distributed over the kinetic and potential energy flows into and out of the reservoir. The throttle controls the valve regulating the total energy flow into the system and the elevator controls the valve distributing the energy flow. Figure 3.1 is a graphical representation of the analogy, showing the energy flows (the single headed arrows indicate positive flows).

3.3.4 Energy awareness and energy management

Although the pilot's intentions are to control the aircraft speed and altitude, in doing so he or she acts on the energy state of the aircraft. However, the pilot can only do this effectively if he or she can identify the energy state.

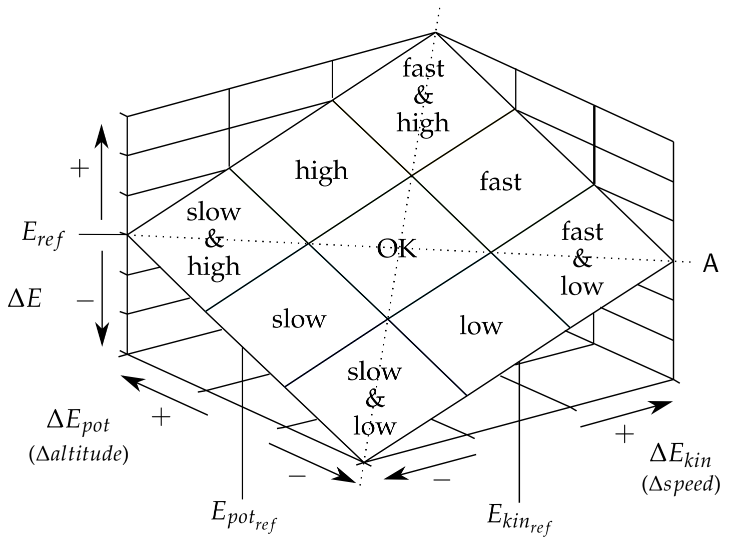


Figure 3.2: The energy state matrix translates speed and altitude deviations into energy deviations. Line A indicates the situation of a correct total energy.

Most pilots have a gut feeling about this, based on their experience, and feel safe when they have lots of energy. They will avoid low–energy states because the lower energy boundaries are deadly. Insufficient kinetic energy means that the aircraft is moving too slowly and is close to a stall. A lack of potential energy means that the aircraft is dangerously close to the ground. The combination of low and slow is especially dangerous because the pilot no longer has the freedom to quickly pull up and gain altitude at the cost of speed to avoid obstacles, or to dive and quickly pick up speed to prevent a stall. The pilot likes to be fast and high, where there is lots of energy to exchange for safe maneuvering. This is a rudimentary form of energy awareness. When the pilot is flying a precision approach, however, he or she will have to be able to identify the energy state much more precisely to use the energy controls to correct deviations from the speed and path goals. It is common that the approach to landing goals are defined as altitude and speed profiles. However, it is also possible to frame the landing goals in energy terms. The target energy path would be a gradual reduction of total energy, so that a suitable energy level is achieved at touchdown. When the pilot is confronted with deviations from the commanded altitude and/or speed he will somehow have to translate those deviations into actions to be taken in terms of energy since the controls are energy controls as shown in the 'reservoir analogy'. This translation can be made by referring to the energy state matrix, Figure 3.2, which shows the possible energy state deviations of the aircraft. On the vertical axis the total energy deviation ΔE is the sum of the kinetic ΔE_{kin} and potential energy deviations ΔE_{pot} on the horizontal axes. Each cell of the grid represents a state deviation from the reference state defined by the total, kinetic and potential reference energies E_{ref}, $E_{pot_{ref}}$ and $E_{kin_{ref}}$ respectively.

Line A is the line of zero total energy error. Cells on this line have the proper total energy but the distribution of the total energy over kinetic and potential energy may be inadequate. The solution to the problem is an exchange of energy that can be realized by using the elevator. Deviations from Line A represent a total energy error: When moving up in the energy matrix the total energy is too high; when moving down in the matrix the total energy is too low. These deviations can only be corrected by, respectively, decreasing or increasing the total energy using the throttle. Generally, the pilot is required to coordinate the controls to bring the aircraft into the right energy state.

Figure 3.2 shows a useful representation for understanding a typical heuristic that some pilots use to solve the approach problem. Many of the pilots that we interviewed reported that they set the throttle to a fixed level (e.g., in terms of engine RPM) at the start of the approach and then use their elevator control to descend along the glidepath with the proper airspeed. If the throttle setting is right, the total energy will decrease at the same rate as the potential energy, which will bring down the aircraft at a constant speed. Any altitude and speed errors (i.e., what we now recognize as energy distribution errors) can be controlled using the elevator. This also means that the errors are correlated; that is, when all goes well, you can tell your altitude error (low or high) from your speed (fast or slow). Taking this line of thought one step further, when the pilot perceives that the altitude and speed errors are not correlated, he or she knows that the throttle setting must be wrong.

3.3.5 Dynamics and cross–coupling

Of course, an aircraft does not have actual valves to control energy as represented in the reservoir analogy. In this section we take a closer look at how

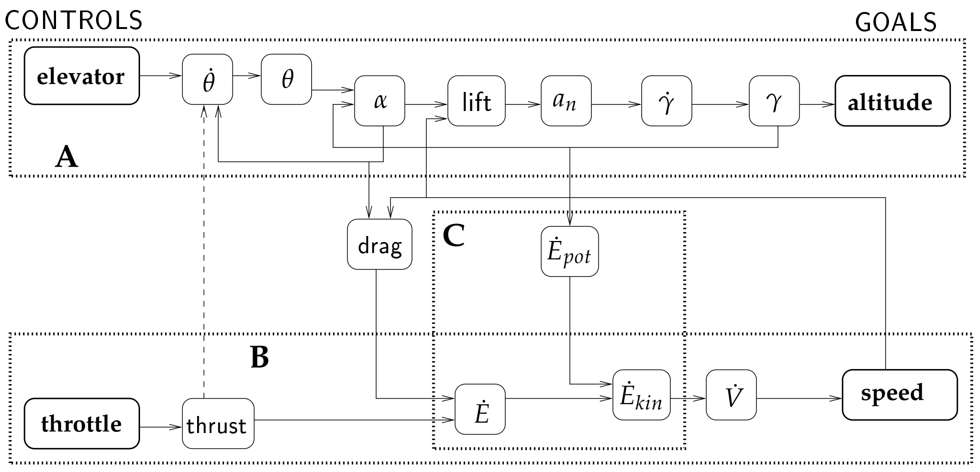


Figure 3.3: The cause–effect relations between the aircraft control manipulators (left) and the pilot's goals (right), characterized through the main aircraft symmetric state variables. These relations reflect the so–called short period approximation of the aircraft vertical flight dynamics (Brockhaus, 1994).

the energy relations are represented in the physical cause–effect relations among the controls, the aircraft state, and the energy reservoirs. Figure 3.3 illustrates how the controls affect the goals through the relevant state variables (Amelink, 2002). The arrows represent causal links between control and state variables (i.e., how one variable can influence the value of another), and these links may include dynamics. It is important to realize that the arrows do not represent energy flows or forces.

There are three main areas of interest indicated by Boxes A, B, and C. Box A shows how elevator inputs directly control the vertical flight–path angle γ and the altitude. Box B shows how throttle inputs lead to speed. This is how novice pilots think of the controls. The complexity comes with Box C, which shows the energy relations and forms the most important link between the direct elevator control path (Box A) and the direct throttle control path (Box B). The link indicated with a dashed arrow represents the cross–coupling of the aircraft's pitching tendency due to thrust changes. This cross–coupling may determine the preference for either control strategy, as discussed at the end of this section.

Our main interest is in the role of the content of Box C. Again the law of conservation of energy tells us that the energy rates must add up. The total energy rate is the sum of the potential and kinetic energy rates:

$$\dot{E} \; = \; \dot{E}_{pot} + \dot{E}_{kin} \; . \tag{3.4}$$

In Box C the relations are drawn for a conventional generic aircraft. Because the thrust acts more or less along the flight path it will first of all accelerate the aircraft so that the total energy added to the aircraft wants to become kinetic energy. This is represented by the arrow connecting $\dot{E}$ and $\dot{E}_{kin}$. However, the elevator can be used to achieve a certain vertical flight–path angle γ that is directly related to the potential energy rate. In other words, the elevator can be used (indirectly, through the aircraft's flight–path dynamics) to demand a certain amount of potential energy rate. The kinetic energy rate is a result of the total energy rate minus the demanded potential energy rate. This is what the arrows in Box C represent. Thus the elevator controls the speed indirectly.

In large transport aircraft, the added total energy due to throttle input tends to become kinetic energy because the coupling represented by the dashed arrow in Figure 3.3 is usually weak. For these aircraft the throttle–to–speed and elevator–to–path strategy may be preferred. Small trainer aircraft, however, have a much stronger coupling as they commonly have the characteristic to pitch up when throttle is applied. Thus, the throttle has a direct effect on the vertical flight path, and therefore also on the potential energy demand, in the same way that the elevator does. Going back to the reservoir analogy, one could say that in this case the throttle also partly operates the energy distribution valve. Hence, for smaller aircraft the throttle–to–path and elevator–to–speed strategy may be preferred.

3.3.6 Short–term and long–term control

From a pilot's perspective, temporal dimensions exist in the control task as well: short–term control and long–term control. The short–term control is the correction of the state deviations.The controls are used for their direct effect as represented in Figure 3.3. Once the state deviations are corrected, the pilot wants to trim the aircraft, creating a steady flight condition at the commanded flight path and speed. This is what pilots call stabilizing the aircraft, and it is referred to as long–term control. The controls are no longer used for direct control but their settings have to be found that lead to the desired steady flight condition. By definition the speed is constant in steady flight. In Figure 3.3 this means that the kinetic energy rate has to be zero and that the total energy rate and potential energy rate must be equal (see Equation 3.4). Thus the throttle has to be set to comply with the commanded vertical flight–path angle. For long–term control the throttle and elevator need to be coordinated such that the elevator controls the vertical flight path and the throttle is used to match the total energy rate to the potential energy rate demand.

3.4 Abstraction hierarchy analysis

The AH has a number of important properties. First of all, each level is a complete representation of the system under consideration. The level of abstraction determines the view on the system and results in a set of terms, concepts, and principles unique to that level. The relation between the levels was described by (Rasmussen et al., 1994) as the why–what–how relation. Each level has this relation with its adjacent levels. For example, looking at the general function level in Figure 3.4 we find energy awareness and energy management. The reason for energy management is defined one level higher, on the abstract function level. The energy is controlled through controlling the right state variables described on the physical function level. The upper levels of the AH describe the goals and the lower levels describe the means available to achieve these goals.

The scope of the analysis defines the top level of the hierarchy, which is the functional purpose of the system. The lowest levels are defined to describe the less abstract, physical implementation of the system's function, the aircraft itself. Therefore the top level and bottom levels are already defined.

The analysis in the previous section allows us to fill in the second (abstract function) and third (general function) levels of the AH. Figure 3.4 shows the content of each level related to the associated part of the analysis. The names of the levels of the AH are adopted from Rasmussen et al. (1994) and their content becomes:

1. *Functional purpose*: The system's meaning to the environment. The goal of the aircraft, considering the task of manually controlling the aircraft longitudinal motion, is to follow the altitude and speed profiles set by the nominal trajectory.

2. *Abstract function and priority measures*: The energy relations govern the aircraft's movement in the vertical plane. This level describes the energy laws

that the aircraft's motion has to obey and centers on the law of conservation of energy. The speed and altitude goals are expressed in energy goals. To satisfy the goals on the level above, the energy goals have to be satisfied.

3. *General function and work activities*: These are independent of the physical implementation. This level contains energy awareness (Figure 3.2) and energy management (Figure 3.1). The throttle is the aircraft total energy control and the elevator is the energy distribution control. The control of energy rate yields control over the aircraft energy state that has to satisfy the levels above.

4. *Physical function and processes, equipment functioning*: This level is dependent on the physical implementation of the system. The above levels hold for a generic fixed–wing aircraft, independent of the type of aircraft. On this level the cause–effect relations of the aircraft–specific characteristics become important. Examples of these characteristics are the pitching due to throttle control (e.g., the dashed arrow in Figure 3.3) and drag variation with airspeed.

5. *Physical form and configuration*: This level contains a description of all aircraft components. It is highly aircraft specific and is not discussed further here.

3.5 Display mapping: energy

The functional purpose in the AH was defined as following a speed and an altitude profile. To provide an intuitive mapping from the aircraft's motion and position to the functional purpose of following an altitude profile, the design is based on a tunnel–in–the–sky display, an egocentric perspective flight–path display that shows the trajectory to be flown in a three–dimensional format (Mulder, 1999). Figure 3.5(a) illustrates the basic tunnel display that is the starting point of the display design discussed here. The aircraft attitude, heading, speed, and altitude are shown through the conventional artificial horizon, the compass rose, and the speed and altitude tapes, respectively. The tunnel geometry shows the desired path (and thus the altitude profile), whereas a flight–path vector (FPV) symbol depicts the direction of the aircraft motion with respect to this path. Hence, when stated in terms of energy, the tunnel geometry shows the potential energy profile, and the FPV shows the actual potential energy rate.

The next two sections describe in detail how the tunnel display can be aug-

LEVELS OF ABSTRACTION		
Functional purpose	fly trajectory:	follow speed profile follow altitude profile

Abstract function	law of conservation of energy	altitude = potential energy
		speed = kinetic energy
		kinetic + potential = total energy

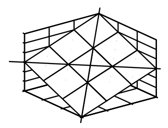
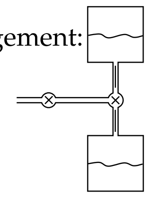
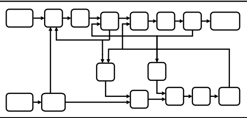
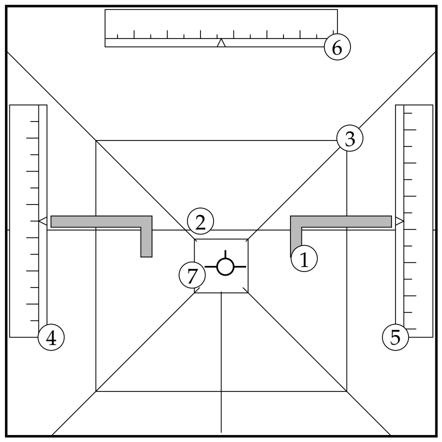
Generalized function	energy awareness: 	energy management: 
Physical function	controlling the state variables	cause-effect of control for a generic aircraft 
Physical form	aircraft-specific components and configuration	

Figure 3.4: The AH for the energy constraints on the task of manually controlling the aircraft symmetric motion.

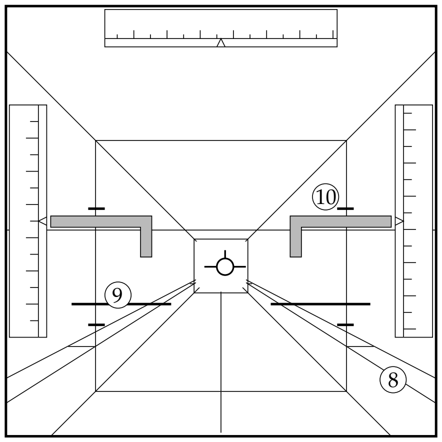
mented with information about energy and energy rate through, respectively, the TERP and the energy angle. Including these two elements in the basic tunnel, as shown in Figure 3.5(b), yields five important new cues for pilots to perceive and act on the aircraft energy state: (a) the total energy deviation, (b) the kinetic energy deviation, (c) a preview of the future nominal total energy state, (d) the total energy rate, and (e) the kinetic energy rate (i.e., the acceleration along the path). In the following sections these five cues are described in detail. Then we discuss how the cues lead to the appropriate control actions.

3.5.1 Expressing energy in a visual format

The concept of the energy–augmented tunnel display is based on predetermined speed and altitude profiles that define the nominal approach trajectory. To visually express energy it needs to be transformed into measures compatible with the tunnel display. A pilot is not very interested in the absolute energy level of the aircraft, but rather in the energy deviations with respect to a target. This way the pilot can use the correction of energy devi-



(a) Generic tunnel–in–the–sky display



(b) Energy–augmented tunnel–in–the–sky display

Figure 3.5: Definition of the various elements and symbols found in a generic tunnel-in-the-sky display *(a)* and the energy-augmented tunnel display *(b)*. In the basic tunnel, the following numbers indicate: (1) aircraft symbol, (2) horizon line, (3) tunnel geometry, (4) speed tape, (5) altitude tape, (6) heading tape, (7) flight-path vector symbol. In the energy-augmented tunnel display, the numbers indicate: (8) the total energy reference plane (TERP), (9) energy angle symbol, (10) speed marks.

ations as the means for achieving the altitude and speed goals. Substituting for the aircraft mass the aircraft weight using the relation: $W = mg$, equations 3.5 and 3.6 give the expression for the potential energy deviation and the kinetic energy deviation:

$$\Delta E_{pot} = W\Delta h \,, \tag{3.5}$$

$$\Delta E_{kin} = \frac{1}{2}\frac{W}{g}(V - V_{ref})(V + V_{ref}) = \frac{1}{2}\frac{W}{g}\Delta V(2V_{ref} + \Delta V) \,, \tag{3.6}$$

with W, g and V as introduced above, h the altitude deviation with respect to the altitude profile as depicted by the tunnel: $\Delta h = h - href$; and ΔV the speed deviation with respect to the speed profile: $\Delta V = V - Vref$. The reader should note that the potential energy deviation is thus defined positive when the aircraft is flying higher than the reference height. The kinetic energy deviation is defined positive when the aircraft is flying faster than the reference velocity. The sum of Equations 3.5 and 3.6 is the total energy deviation:

$$\Delta E \;=\; \Delta E_{pot} + \Delta E_{kin}\;, \tag{3.7}$$

In the tunnel display the potential energy deviation is already present in the form of height; it is the aircraft vertical deviation from the tunnel centerline. In this respect, the tunnel serves as a commanded potential energy profile. Then, through expressing the kinetic energy deviation relative to this height the energy representation can be completed (Figure 3.6). This is accomplished by dividing Equations 3.5 and 3.6 by the aircraft weight W, yielding the kinetic energy deviation height:

$$\Delta h_{E_{kin}} \;=\; \frac{\Delta E_{kin}}{W} \;=\; \frac{1}{2}\Delta V \frac{2V_{ref} + \Delta V}{g}\;, \tag{3.8}$$

and the total energy deviation height:

$$\Delta h_E \;=\; \frac{\Delta E}{W} \;=\; \Delta h + \frac{\Delta E_{kin}}{W} \;=\; \Delta h + \frac{1}{2}\Delta V \frac{2V_{ref} + \Delta V}{g}\;, \tag{3.9}$$

When the approach trajectory is defined by an altitude and speed profile, the potential, kinetic and total energy profiles are now implied. Figure 3.6 illustrates how the energy relations are represented using the aircraft position, the tunnel centerline and the TERP, which is constructed by subtracting the kinetic energy deviation height from the commanded height (the tunnel centerline height h_{ref}).

Figure 3.7 illustrates the relations among the aircraft, the tunnel trajectory, and the TERP, according to the energy state matrix of Figure 3.2. The center picture shows the desired situation where the aircraft is flying along the tunnel centerline at the right speed. Some important properties can be noticed. First, the aircraft height above the tunnel centerline represents the positive potential energy error; the aircraft height below the tunnel centerline represents the negative potential energy error. When the aircraft is aligned with the tunnel, the potential energy error is zero. Second, the aircraft height above the TERP represents the positive total energy error; the aircraft height below the TERP represents the negative total energy error. When the aircraft is aligned with the TERP, the error in total energy is zero. Third, the vertical separation between the tunnel centerline and the TERP represents the error in kinetic energy: When the TERP moves below the tunnel centerline, the aircraft is flying too fast, and when the TERP moves above the tunnel centerline, the aircraft is flying too slowly. Note that this property

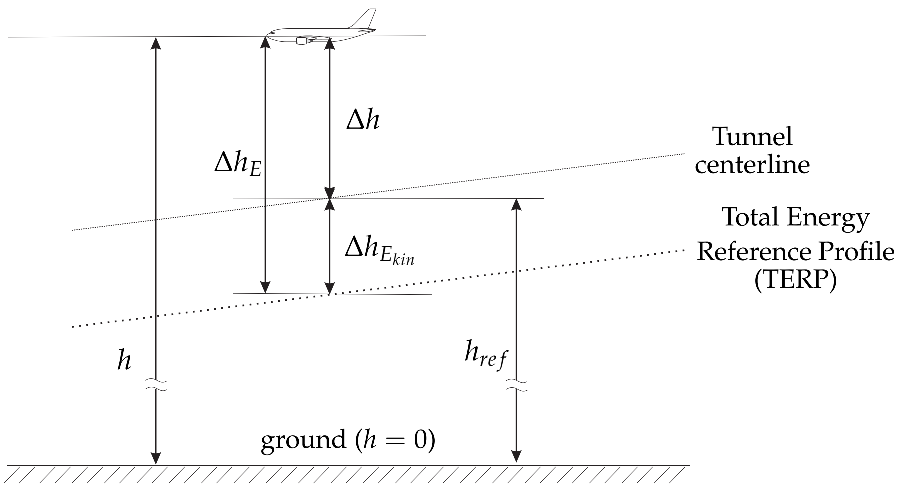


Figure 3.6: The Total Energy Reference Profile (TERP) is based the concept of expressing energy deviations in height. In this figure, the aircraft is flying above the tunnel centerline (high: $\Delta h > 0$, $\Delta E_{pot} > 0$), indicating the positive potential energy deviation. It has a speed that is higher than the reference speed (fast: $\Delta V > 0$, $\Delta E_{kin} > 0$); that is, a positive kinetic energy deviation as reflected by the fact that the TERP has moved below the tunnel centerline. The vertical distance between the aircraft and the TERP indicates the total energy deviation.

is independent of the position of the aircraft relative to the tunnel. Again, similar to Figure 3.2, Line A shows the situation where the total energy error is zero (the aircraft is aligned with the TERP) but there can be an energy distribution error (so the pilot can exchange energies to get to the desired energy state, through the elevator).

3.5.2 Visual design

The challenge for designing the visual display was to configure the representation of the TERP with the three–dimensional tunnel display. This was accomplished by using linear perspective relations associated with presenting the TERP as an energy surface that emerges parallel to the nominal trajectory. In Figure 3.8a a top–down view on the geometrical design of the TERP representation is shown. Lines (*i*) are the inner path lines, which coin-

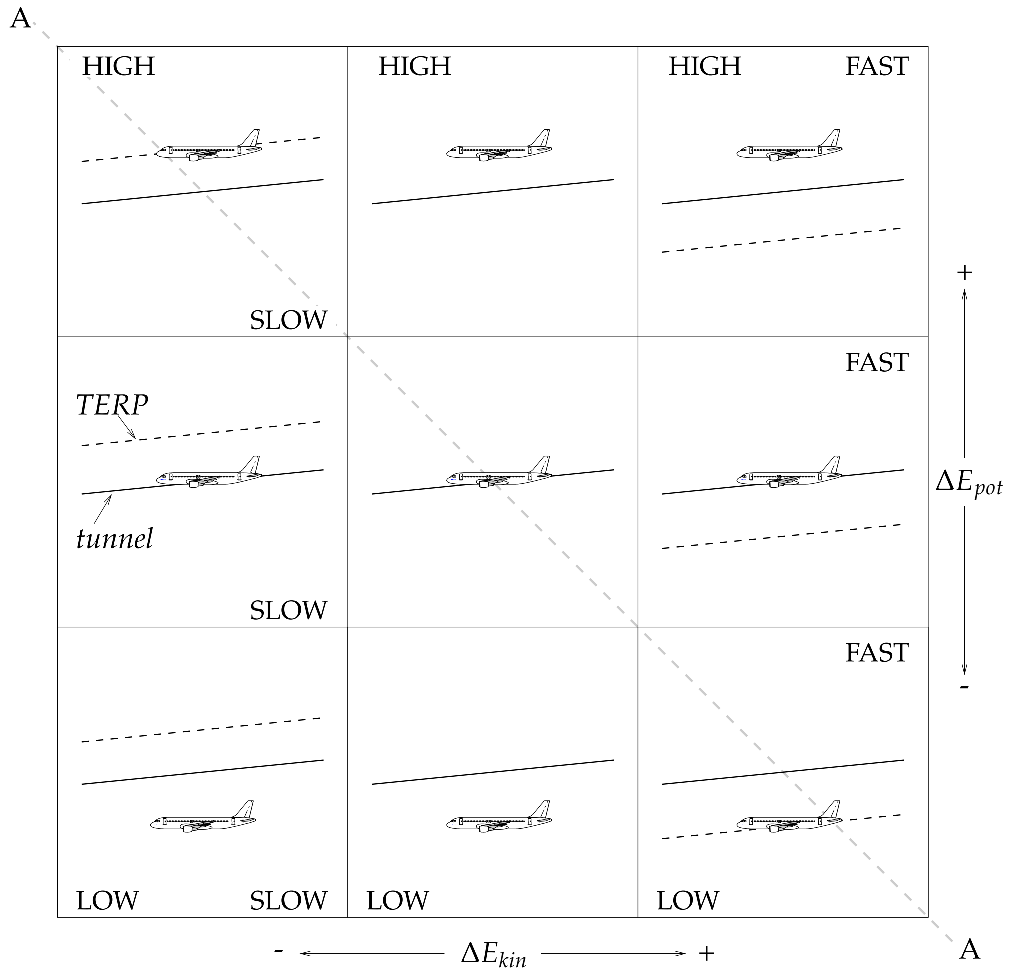


Figure 3.7: The energy state matrix of Figure 3.2, now defined with respect to the reference potential and kinetic energy states, indicating how the relative positions of the aircraft, the tunnel centerline (dashed lines), and the TERP (thick gray lines), reflect the energy deviations.

cide with the tunnel sides. Lines (*ii*) are added texture marks that coincide with the lateral position of the tunnel frames. These marks are expected to help the pilot judge the vertical distance between the tunnel and the TERP (for the perception of the speed deviation). Lines (*iii*) are the outer path lines and basically connect the endpoints of the texture marks to make it a more cohesive representation.

This representation is defined to visually imply a surface while using only the space outside of the tunnel to avoid clutter in the center of the display. Figure 3.8b shows the TERP as shown on the perspective display, clearly illustrating that when the aircraft is high on energy, the TERP is below the aircraft (below the viewpoint); when the aircraft is low on energy the TERP is viewed from below (the TERP has moved above the viewpoint); and when the aircraft has a zero total energy error, the TERP is perfectly aligned with the aircraft and its projection on the display reduces to a line.

3.5.3 Perceiving the energy cues from the energy augmented tunnel display

The surface analogy is a fundamental property of the energy–augmented tunnel display. In a comprehensive study, (Mulder, 1999, 2003) showed that surfaces play a crucial role in understanding the way pilots perceive the aircraft locomotion state from the three–dimensional tunnel geometry motion perspective. Motion relative to a surface yields an optical expansion pattern that contains very useful information about the observer's motion (Gibson, 1950, 1986; Flach, Hagen, & Larish, 1992): the texture gradients. Visualizing energy through a surface, the TERP, aims at enabling pilots to directly perceive the aircraft energy state through the texture gradients resulting from the aircraft motion relative to the energy surface (Amelink, 2002).

Potential energy deviation

The aircraft potential energy deviation is represented by the common splay angle and density texture gradients that emerge when the tunnel geometry changes due to changes in the vertical aircraft position relative to the tunnel centerline (Mulder, 1999, 2003).

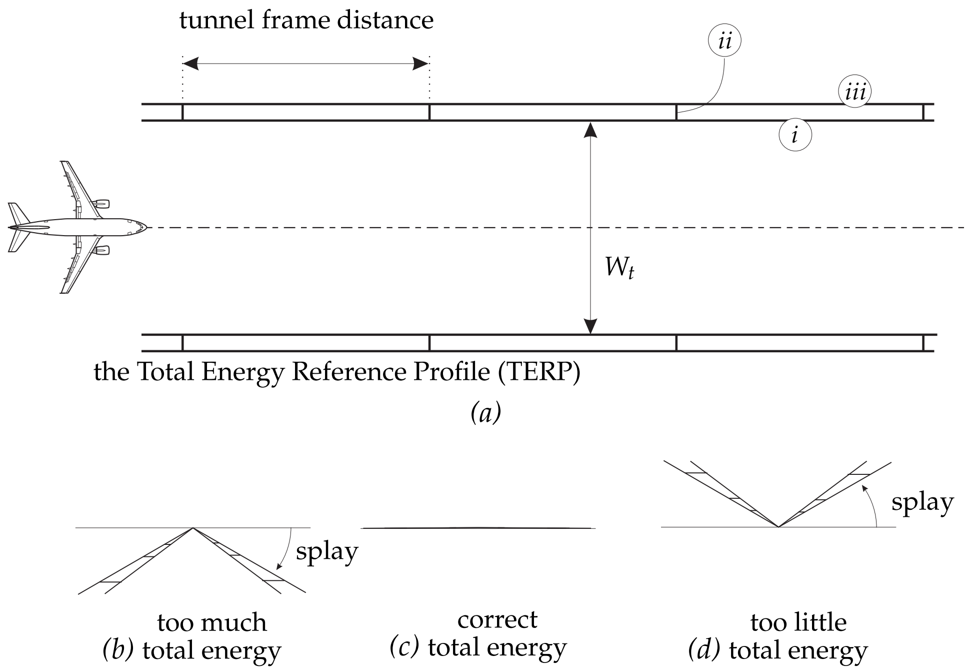


Figure 3.8: The visual design of the TERP. The top figure shows a God's-eye view on the TERP. The bottom figures show how the TERP geometry changes when projected on a perspective display when the aircraft is flying either above (left), below (right), or in alignment with the TERP (middle). W_t is the tunnel width.

Total energy deviation

The total energy deviation is the first new cue that emerges from the energy–augmented tunnel display. With the surface metaphor, the total energy error is represented as a virtual eye position relative to the energy surface. Figure 3.9 shows the tunnel display augmented with the TERP for the same conditions as in Figure 3.7. When there is zero total energy error, the diagonal A–A in Figure 3.9, the eye is at the surface level and the edges become parallel with the horizon. When there is positive total energy error, the top–right drawings, then the eye is above the energy surface and the edges splay out below the horizon. As the total energy error increases, the edges become more and more splayed away from the horizon (in a way that is similar to the edges of a roadway beneath an aircraft). When there is a negative total energy error, the bottom–left drawings, the eye is below the energy surface

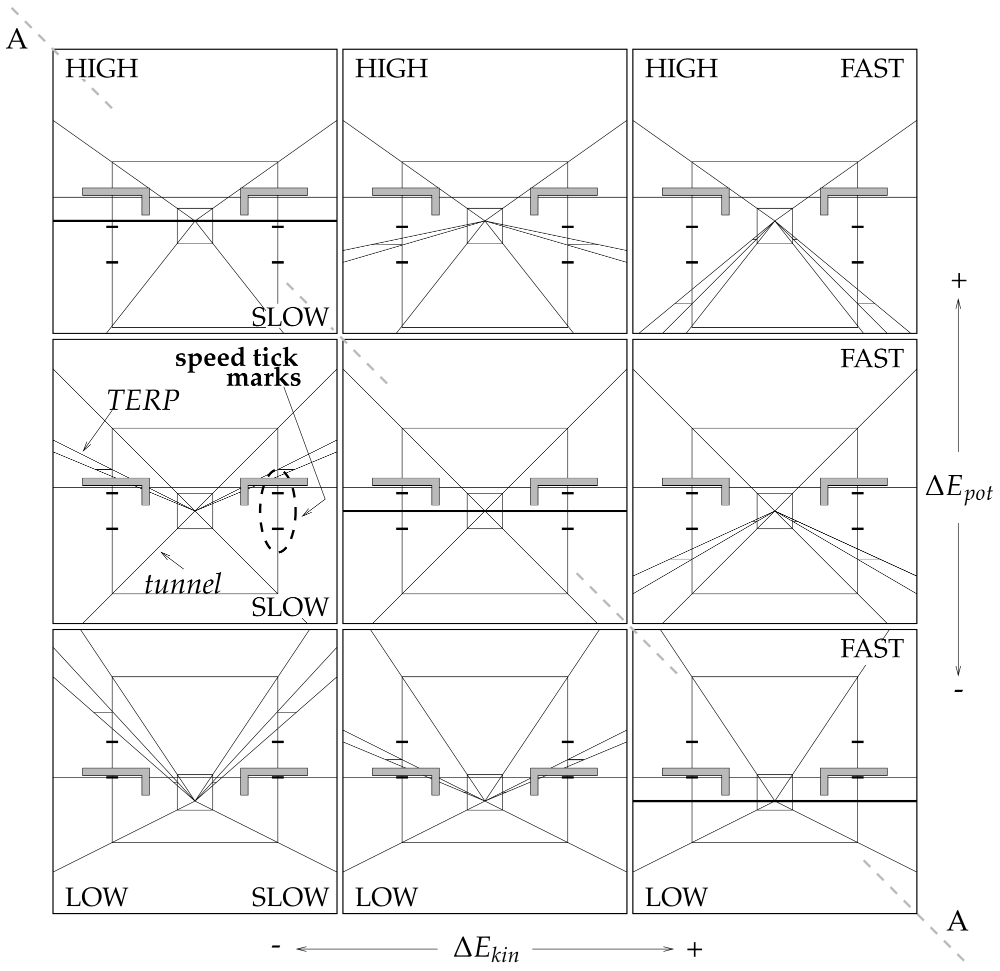


Figure 3.9: The energy state matrix of Figure 3.2 (and Figure 3.7), now shown through including the three-dimensional TERP surface (defined in Figure 3.8) into the perspective tunnel display.

and the edges splay out above the horizon (in a way similar to tiles on the ceiling). The total energy deviation is the sum of the potential energy deviation and the kinetic energy deviation (Equation 3.7), and these are represented by the position of the aircraft and the TERP relative to the tunnel, respectively.

Kinetic energy deviation

This is our second new cue. The aircraft speed deviation ΔV is represented by the vertical separation between the tunnel and the TERP. This distance, the kinetic energy deviation expressed in height, is the cue for speed deviations, independent of the aircraft position relative to the tunnel. When the speed is right, the total energy error equals the potential energy error and the aircraft height above the tunnel centerline equals the height above the TERP. This is illustrated in the drawings on the center column in Figures 3.7 and 3.9. Due to this property, the allowable speed deviations with respect to the reference speed profile can be projected on the tunnel sides through the speed tick marks, reflecting the fact that speed deviations are visualized through the vertical separation between the TERP and the tunnel centerline (Figure 3.9).

3.5.4 Preview of the future required energy state

An important aspect of the energy–augmented tunnel display is that it provides a preview of the future commanded energy state, our third new cue. The pilot can actually see if the stabilized condition will take the aircraft to the commanded current or future energy states, which should yield a much better anticipation. The other advantage of a preview is that the pilot can see a commanded change of path or speed well in advance. Figure 3.10 shows two types of changes that can be encountered. A required speed change, as Figure 3.10a shows, can be recognized by the upcoming vertical separation of the tunnel and the TERP. A flight–path change can be recognized by an equivalent change of TERP and tunnel shown in Figure 3.10b. As these changes can also take place simultaneously, the display still shows the pilot a valid energy representation.

When the aircraft energy state is not correct, or, equivalently, when its altitude and speed do not match the reference profiles, the pilot needs to manipulate the controls that are available to control the total energy (throttle)

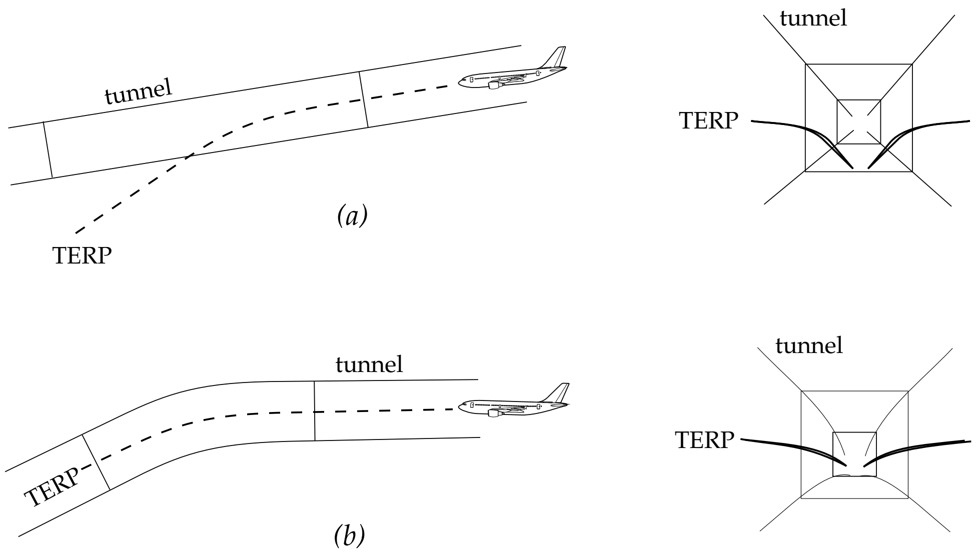


Figure 3.10: The energy-augmented tunnel-in-the-sky display provides the pilot with preview of *(a)* a commanded deceleration and *(b)* a commanded descent. The preview is the third new cue conveyed by the energy-augmented display.

and the energy distribution (elevator). Then, when changing the total, potential, and kinetic energies, it is mandatory to have an indication of the energy rates. The mapping of energy rate on the display is the subject of the next section.

3.6 Display mapping: energy rate

3.6.1 Expressing energy rate in a visual format

After identifying the energy deviations, a pilot is concerned with correcting them using elevator and throttle, controlling the energy rates. At this stage, energy rates are the means to correct any energy level deviations (i.e., short–term control). Note that the potential energy rate is already present in the tunnel–in–the–sky display, as it is depicted by the position of the FPV symbol with respect to the horizon line. The vertical flight–path angle γ is referred to as the aircraft–specific non–dimensional potential energy rate

(Lambregts, 1983a). Similarly, the total energy rate can be expressed in the total energy angle γ_E (in the following referred to as the energy angle):

$$sin\gamma_E \;=\; \dot{E}_{sn} \;=\; sin\gamma + \frac{\dot{V}}{g}\,. \tag{3.10}$$

In this equation, $\dot{E}_{sn}$ is the aircraft specific non–dimensional total energy rate and is the aircraft acceleration along the flight path. For small flight angles γ the following approximation is valid:

$$\gamma_E \;=\; \gamma + \frac{\dot{V}}{g}\,. \tag{3.11}$$

This relation expresses all energy rates in angles, which makes it compatible with the tunnel display. As discussed further later, showing the energy angle in conjunction with the already present FPV symbol reveals the three energy rates to the pilot. The energy angle is also known as the potential flight path; that is, the flight–path angle that will maintain the existing airspeed based on the current thrust and drag (Brockhaus, 1994). Showing the potential flight path is common for HUDs, particularly in the landing phase (Newman, 1995), and has been applied before in a perspective format (Theunissen & Rademaker, 2000).

3.6.2 Visual design

The way in which the flight angles are represented by symbols in the display is illustrated in Figure 3.11. The vertical distances between the symbols and the horizon represent the angles: The aircraft pitch attitude θ is the distance between the fixed aircraft symbol and the horizon. The distance between the horizon and the FPV indicates the flight path γ. The distance between the energy–angle symbol and the horizon indicates the potential flight path γ_E. The form of the latter symbol is a long horizontal line with a gap in the middle, to prevent overlap with FPV. Because the energy angle does not have meaning in the lateral plane the line is always parallel to the horizon and the gap is always aligned with the FPV. As becomes clear later, the line needs to be long to have an overlap with the TERP.

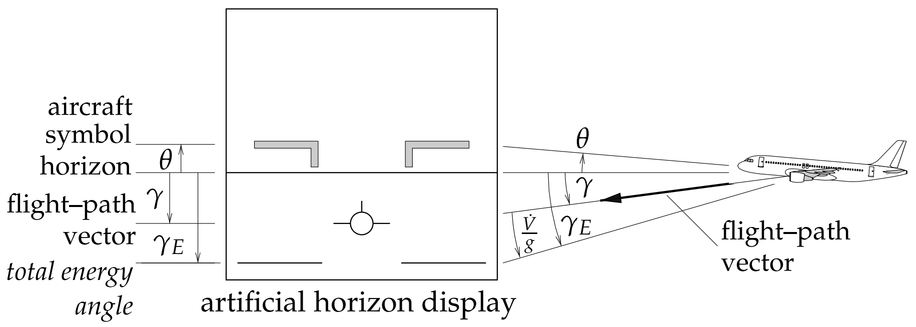


Figure 3.11: The display symbols representing the aircraft attitude, flight-path and energy-angles. In this figure, the flight-path vector symbol is below the horizon, indicating a descent (γ is negative: decrease of potential energy); the total energy angle symbol is also below the horizon, indicating a reduction of total energy (γ_E is negative). The flight-path vector symbol is positioned above the total energy angle symbol, indicating a deceleration (decrease in kinetic energy).

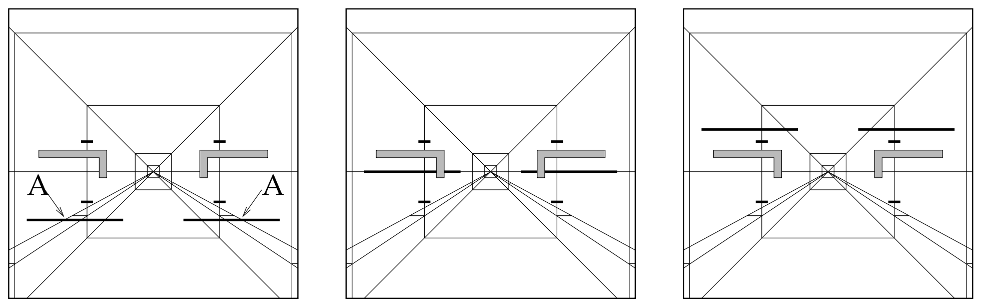
3.6.3 Perceiving energy rate cues from the energy augmented tunnel display

Potential energy rate

The potential energy rate is in fact shown by the FPV symbol, a very common representation in artificial horizon displays. When the FPV is aimed below (or above) the horizon, the aircraft descends (or climbs) and the potential energy decreases (or increases). When the FPV is put on the horizon line, the potential energy remains constant. When the FPV is aimed at the vanishing point of the tunnel, the potential energy rate equals the potential energy rate as required by the nominal trajectory.

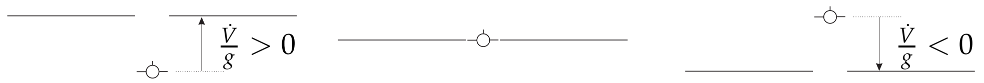
Total energy rate

The fourth new cue presented in the energy–augmented tunnel display is the total energy rate, which can be expressed in the energy angle γ_E, defined relative to the horizon. When the energy angle is above the horizon there is a total energy increase; when it is below the horizon there is a total energy decrease. Because the throttle is the total energy valve, the total energy angle can be moved up and down using the throttle. The energy



(a) converging to the TERP *(b)* parallel to the TERP *(c)* diverging from the TERP

Figure 3.12: The intersection of the energy angle symbol and the TERP in the display indicates the point of interception (point A) of the TERP in the future. The energy angle symbol shows the total energy rate.



(a) acceleration *(b)* steady state *(c)* deceleration

Figure 3.13: The difference between the energy angle symbol and the flight-path vector symbol represents the acceleration along the flight path. This is the fifth and final new cue, presenting the kinetic energy rate.

angle also represents the energy flight path to the TERP. This is illustrated in Figure 3.12: When there is an overlap between the horizontal lines of the energy angle symbol and the TERP, the total energy is converging to the commanded total energy level. The point along the TERP where it is intersected by the energy angle symbol is the point where the aircraft intersects the TERP for the given throttle setting.

Kinetic energy rate

The fifth and last new cue is the kinetic energy rate cue. Figure 12 illustrates that the energy angle γ_E is the sum of the vertical flight–path angle γ and the non–dimensional acceleration $\dot{V}/g$ (Equation 3.11). Thus, the difference between the energy angle and the FPV expresses the acceleration along the flight path. Figure 3.13 shows that when the FPV symbol is below the energy angle symbol the aircraft accelerates ($\dot{V}/g > 0$), when it is above the energy

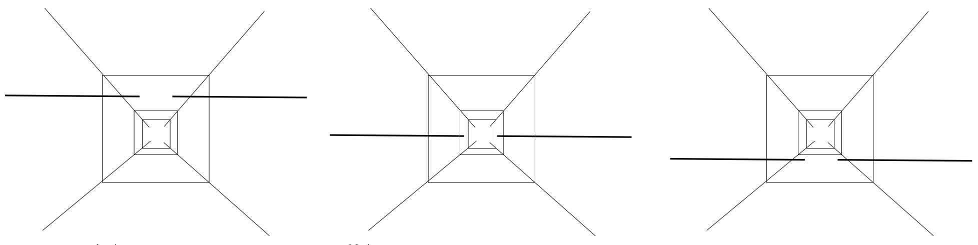


Figure 3.14: The relative position of the energy angle symbol and the tunnel geometry indicates the throttle setting for a steady flight path. In a stationary flight *(b)*, the flight-path vector symbol but also the energy angle symbol must be aligned with the tunnel geometry, otherwise the aircraft will accelerate *(a)* or decelerate *(c) because the throttle setting is too high or too low, respectively.*

angle symbol the aircraft decelerates ($\dot{V}/g < 0$), and when both symbols are aligned the speed remains constant ($\dot{V}/g = 0$). Note that one can accelerate along the flight path with the elevator control by putting the FPV symbol below the energy angle, and also with the throttle control, by putting the energy angle above the FPV. Similarly, to decelerate along the flight path the same opportunities exist. It is clear that this cue is important for finding and maintaining a steady flight condition: The aircraft establishes a constant speed only when the FPV and the energy angle symbols are aligned.

The energy angle is also known as the potential flight path; that is, it shows the vertical flight–path angle that the current throttle setting could sustain at the current speed (Newman, 1995). When stabilized (i.e., a constant speed), the energy angle symbol, like the FPV, has to match the direction of the future tunnel trajectory, as illustrated in Figure 3.14b. Hence, it directly links the throttle setting to the commanded flight path: It lets the pilot set the throttle independent of the momentary flight–path angle and speed in a very direct way.

3.7 Working with the Cues

In the previous two sections we showed that the energy–augmented tunnel–in–the–sky display, including the TERP and the energy angle, contains five important new cues for the manual control task. The cues show energy deviations and the means to correct them. The throttle, through the engine dynamics, controls the energy angle γ_E, and the elevator, through the aircraft dynamics, controls the vertical flight path γ. In other words, the energy angle symbol and the FPV symbol can be manipulated with the throttle and the elevator, respectively.

How the cues are exactly used during flight can only be evaluated experimentally. However, the following can be said based on discussions with experienced pilots and the analysis of the task (Amelink et al., 2003a). The task of piloting can be split into long–term control and short–term control. The short–term control is concerned with the immediate response of the aircraft used to correct deviations and to follow the flight path and speed profile. The long–term control is what pilots call stabilizing. It is concerned with balancing the forces that act on the aircraft so that it will naturally fly at the commanded speed and vertical flight–path angle in steady state. The cues in the display present information to the pilot that facilitates both parts of the control task.

3.7.1 Short–term control

For short–term control, the deviation from the commanded energy state should be directly perceived. The aircraft vertical position relative to the tunnel centerline is the cue for the potential energy deviation. In the same way, the aircraft vertical position relative to the TERP represents the aircraft total energy deviation. Now, there are four possibilities, considering the aircraft to be in steady state each time. First, the aircraft is on the tunnel centerline and on the TERP. This represents the commanded energy state and there is no need for corrections. Second, the aircraft is on the tunnel centerline but not on the TERP. In this case there is not a potential energy error and the kinetic energy deviation equals the total energy deviation. To correct the total energy error the pilot must use the throttle. Third, the aircraft is on the TERP but not on the tunnel centerline. There is not a total energy deviation but an energy distribution deviation. The kinetic and potential energy deviations are equal but opposite. Elevator control should be used to correct the deviation. Fourth, the aircraft is on neither the tunnel centerline nor the TERP. Both total energy and energy distribution deviations occur. Both throttle and elevator are needed to correct the deviations.

In all cases, the pilot uses feedback from the energy angle symbol and the

FPV symbol to control the energy rates with throttle and elevator. They show the angles at which the TERP and tunnel will be intercepted, respectively, allowing pilots to precisely select the control inputs to correct the deviations and to trim the energy rates for the long–term control.

3.7.2 Long–term control

Matching the energy rates to the commanded flight path and speed is the key to stabilizing the aircraft. When stabilizing, the energy angle is used in conjunction with the tunnel to determine the stabilized throttle setting; the energy angle should be aligned with the future tunnel center (as in Figure 3.14b). To obtain a stabilized, stationary flight along the tunnel, the FPV symbol as well as the energy angle symbol should be aligned with the tunnel geometry. This correlates with the acceleration cue shown in Figure 3.13b; speed is constant, thus the acceleration along the flight path, $\dot{V}/g$, is zero.

By showing the total energy angle symbol in relation to the tunnel geometry, pilots obtain a clear insight into what throttle setting is needed to bring the aircraft into a stationary flight condition along the tunnel centerline. Hence, the pilot heuristic mentioned earlier, where pilots fix the throttle to a certain setting and then control the approach with elevator only, is very likely to be replaced by (many) new rules for setting the throttle. Furthermore, pilots will obtain a good understanding of why these rules work.

3.8 EID related properties of the display

EID (Vicente & Rasmussen, 1992) is a theoretical framework for designing interfaces for complex human–machine systems. It is the approach to interface design that gives priority to the worker's environment, concentrating on how the environment imposes constraints on the worker. It is based on the three levels of the skills, rules, and knowledge (SRK) taxonomy (Rasmussen, 1986). A display based on the EID principles should support the operator on all three levels of cognitive processing. How the three levels are supported in the energy display is discussed next.

Skill–based behavior is based on time–space signals that can be directly used for control. The five cues previously discussed all represent signals, and they are all compatible with skill–based behavior. This can be illustrated by the following: When a hypothetical pilot is completely unaware of the energy constraints, he or she should be able to fly the approach by just keeping the energy angle on the TERP with the throttle and the FPV symbol inside the tunnel with the elevator. This is a skill–based tracking task with a low cognitive load.

Rule–based behavior is based on the perception of signs in the work domain that trigger a set of previously stored rules for dealing with a familiar situation. The two expected changes are speed and glide slope changes (Figure 3.10). Because the perspective trajectory gives a preview of the future commanded energy state, the pilot should be able to recognize, without reasoning, which of the changes is coming up and act on it directly. Also, the tunnel size and the markers on the tunnel wall for the TERP reference height, combined with the ego position in the tunnel and the TERP height, provide the pilot signs about acceptable performance. It is very likely that with the energy display new rules will emerge to replace the existing pilot heuristics and control strategies of throttle–to–path, elevator–to–speed and the like. Also, pilots will learn quickly how the relations between TERP and tunnel (Figure 3.9, reflecting the energy matrix of Figure 3.2) relate to elevator and throttle settings, resulting in new behavioral patterns as a result of the new rules. In other words, pilots will recognize the various energy deviations and develop cognitive short–cuts to efficiently and effectively deal with them.

Knowledge–based behavior is based on the perception of symbols that carry meaningful information in the work domain used for unanticipated situations and problem–solving activities. The visualization of the energy constraints is based on the top three levels of the AH and should allow for reasoning and problem solving. The visualization does not tell the pilot what to do, but it shows the structure of the energy constraints revealing possible solutions. The pilot is allowed to choose any control strategy that satisfies the system goals. This results in a naturalness of control that is not available from interfaces based on a more conventional, procedural task analysis.

It is also possible that the energy display could serve well as a teaching tool, to explain to pilots how the energy balance works, how it governs the aircraft vertical motion, and how the available elevator and throttle controls can be used effectively to achieve the goals. In this way, pilots will get a much deeper understanding of what is really happening and why certain strategies work.

3.9 Final Remarks

The derived AH is a representation of the energy constraints as a subset of the complete aviation work domain. It can serve as an externalized mental model that an experienced pilot has for the symmetric aircraft control. As Vicente and Rasmussen (1992) stated, the AH can only represent what the designer, researcher, or expert knows. It has not come up with answers about the energy constraints that were unknown, but the AH framework has helped to structure the process of analyzing the control task. It enabled us to ask the right questions about the work domain and structure it in a psychologically relevant way that supports the mapping of goal–directed behavior.

The next step was the actual design of a display to present the energy information to the pilot. The AH is part of the EID framework (Vicente, 2002) applied to the design of the energy display. The energy display is the result of a comprehensive study of the energy constraints in flight using the AH and EID principles (Amelink, 2002). Whether the contributions of the EID design are beneficial in terms of the common metrics like pilot performance, workload, and situation awareness is unclear. Our initial subjective evaluation with professional pilots, however, looks promising. Initially, pilots found the displays to be rather complex. This comment was not unexpected. An ecological interface must capture the requisite variety of the domain, so when the domain is complex this will be reflected in the interface. To quote Tufte (1990), "to clarify: add detail" (p. 37). With experience, however, pilots learned to see and use the energy constraints and the final comments were very positive.

Research is underway to investigate the energy–augmented display through an extensive pilot–in–the–loop evaluation. We hypothesize that whereas performance levels may well be unaffected by the display, the whole nature of the activity will change significantly, and that the display will allow pilots to explore other control strategies, increasing their flexibility in the kinds of approaches they may conduct. In particular the use of the throttle will change. The current heuristic of setting the throttle before the approach and continuing to control the landing with elevator reflects the fact that pilots because they do not see what the throttle does, simply lock out this degree of freedom. With the energy display, a much more frequent use of the throttle is expected because pilots can now directly see how the throttle setting is related to the speed and altitude goals, enabling them to use their controls in a truly coordinated fashion. Empirical evaluations of the display will include dependent measures that index the amount of coupling across levels of the SRK taxonomy and AH, along the line of thought advocated in Yu, Lau, Vicente, and Carter (2002).

Another direction that we are currently investigating is to consider other contexts in which pilot energy awareness might be particularly important, such as landings under wind shear conditions where large shifts in energy might occur, and tasks such as terrain following, where pilots have to operate their vehicle near the edges of the energy envelope.

4

Case study: Analysis of the Total Energy Control System

In the previous chapter, Work Domain Analysis (WDA) has been used to identify the role of energy management in the manual longitudinal flight control task. It was represented as an intermediate control goal at the abstract function level of the abstraction hierarchy. The analysis was part of Ecological Interface Design (EID) of a novel display that allows pilots to see and act directly on the aircraft energy state, and possibly adapt their control strategies to more effective ones.

To investigate the applicability of WDA to automation design as part of ecological automation design (EAD) as outline in Chapter 1, we study the control problem of Chapter 3 but now as part of an automatic flight control system: The Total Energy Control System (TECS). In contrast to conventional autopilot designs, TECS is based on energy management principles. TECS was developed by Lambregts in the 1980s (Lambregts, 1983a, 1983b) to overcome a number of limitations of conventional flight control system designs. The design of TECS started with a analysis of the fundamental physics of flight. As will become clear, TECS can be considered as an example of an ecological approach to automation design *avant la lettre*. By performing WDA for TECS, we aim to discover how WDA can uncover the principles that have led to the claimed performance increase and reduction of complexity in the system.

First, an introduction to TECS is given to explain the main design philosophy of TECS and the basic architecture. Then, TECS is discussed in detail and the functions of the components are mapped onto the abstraction hierarchy using 'abstraction' and 'aggregation' relations. The chapter concludes with making a first step towards a framework to capture a larger system boundary than the analysis of Chapter 3 and the nested structure of the control problem. In order to be able to link parts of the design to the abstract functions and the functional purpose, we will have to view TECS in a considerable amount of detail. Yet, despite the many system specific details, the focus of this chapter lies on the process of WDA.

4.1 Introduction

The available knowledge on the functioning of TECS is mapped onto the abstraction hierarchy. Therefore, this chapter contains many details on TECS that are first explained and then represented at the various levels of abstraction. Throughout the explanations, the why–what–how relationship that links concepts at different levels of abstraction is visible. TECS is not simplified from the original descriptions to avoid the pitfall of simplifying for the sake of representation of the system in the abstraction hierarchy. However, we cannot include all the details. This chapter also serves as an explanation of the design of TECS, focussing on the reason for design features through abstraction.

4.2 Overview of TECS

Total Energy Control System (TECS) is a generalized automatic flight control system that was developed in the late 1970's to early 1980's by Lambregts, to overcome a number of issues with conventional autopilots at the time (Lambregts, 1983a, 1983b, 1996). These issues include: unnaturally high levels of control activity (especially for the autothrottle), a complex man–machine interface, and functional overlap in control modes. The objectives of his work were to integrate all longitudinal autopilot and autothrottle control functions to generate pilot–like control, to create a simplified man–machine interface, and to structure the control mode hierarchy to eliminate the overlap in control modes of conventional autopilots and auto–throttle

systems.
Lambregts' design approach started with an analysis of the fundamental physics of flight, resulting in a representation of how flight is constrained in terms of energy. Consequently, his analysis of the work domain led to identifying energy management as the *primary* organizing principle for longitudinal aircraft control. This is the main point that distinguishes TECS from conventional autopilot designs, that are organized around the theory of aircraft flight dynamics, and capture the dynamics using linear models that describe small perturbations from steady states. In addition, Lambregts fully recognized the importance of designing with the least amount of complexity and came up with an *all encompassing system design strategy*. As a result, TECS was reported to overcome many of the design deficiencies found in conventional autopilots at that time.
Because of Lambregts' departure point, TECS can be regarded as an ecological approach to automation design "avant la lettre". Since TECS has been designed, implemented, and evaluated in real aircraft, it can teach us valuable lessons on how the ecological approach can guide the design process of automation.
The elimination of unnecessary complexity through systematic redesign of the autopilot/autothrottle architecture is the larger goal of the design of TECS. It was achieved by meeting multiple design objectives of which energy management is only one. Below, the overall approach is discussed to sketch the larger picture, starting with the problems with conventional autopilot/autothrottle systems that motivated the design of TECS.

4.2.1 TECS design goals

The motivation behind TECS was to improve the deficiencies in conventional Flight Guidance and Control (FG&C) systems at the time TECS was designed. The deficiencies and associated goals of TECS are discussed. Conventional FG&C systems typically consist of three main subsystems: the flight control computer, the autothrottle and the navigation/performance computer. With each generation of aircraft, new capabilities were added to these systems. However, the bottom–up approach of adding one flight control level at a time resulted in a non–optimal system architecture, and the three computer systems have considerable overlap in their function. The result is a fragmented system architecture that does not allow for an efficient implementation of general safety provisions such as speed protection and normal acceleration limiting.

Illustrating this with the history of the development of flight deck automation, Lambregts (1996) showed that current systems are the result of a step by step approach, where each control mode and function was a logical addition to expand the capabilities of the automated flight deck or a fix for certain deficiencies. The resulting legacy system suffers from unnecessary design complexity, whereas state–of–the–art systems design would permit the design to be simplified considerably. Lambregts stressed that the automation of flight control functions has greatly contributed to the efficiency and safety of air transport but that the current solutions are sub–optimal. The unnecessary complexity implies unnecessary cost attributed to design, development, operations, maintenance, spares, and training.

The main problems that Lambregts identified are: bad formulation of requirements, adhering too long to outdated technologies, engineering design concepts and processes, and possibly the tendency to adhere to outdated and flawed design concepts for new generations of aircraft while the deficiencies are addressed in a *band–aid* approach. Lambregts called for a redesign of the system architecture to eliminate the unnecessary complex systems that flight control automation has historically evolved into. This is a case where the complexity of the system and its impact on the operators cannot be solved at the man–machine interface alone (Lambregts, 1996).

Lambregts' main objective was to devise a methodology for designing a generic elevator and thrust command computation algorithm, providing decoupled flight path and speed maneuver control and capable of serving all flight path and speed control modes. A second objective was to overcome the discussed limitations of the separately designed autopilot and autothrottle systems (Lambregts, 1983b).

Control decoupling

Historically, the control of the aircraft vertical flight path was assigned to the elevator, and the control of its speed was assigned to the throttle as two separate control loops. This single input–single output (SISO) controller design does not address the coupling between speed and path control. The design, therefore, fails to anticipate required thrust changes with vertical flight path changes. In other words, it ignores the exchange of kinetic energy (speed) and potential energy (altitude). Figure 4.1 outlines the conventional SISO autothrottle and autopilot design.

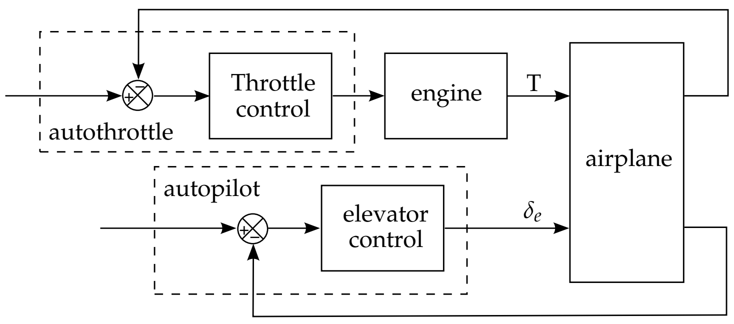


Figure 4.1: Conventional SISO control strategy where throttle and elevator are controlled by two separate control loops (adapted from Lambregts (1983a)).

In this design, path control manoeuvres induce speed perturbations and speed control manoeuvres induce path perturbations. For example, when a path command increased flight path angle, the aircraft will initially slow down before the speed control loop will increase throttle to maintain speed. These control coupling errors and associated controller transients are qualified as un–pilot–like and undesirable. A prevailing pilot complaint about conventional speed control autothrottle systems has been that its operation is often contrary to the way the pilot would handle the throttle control (Lambregts, 1996).

The TECS core implements a multi input–multi output (MIMO) controller, as will be discussed in 4.2.3, to manipulate elevator and throttle based on vertical path and acceleration demands. This allows an energy management control strategy similar to how pilots handle longitudinal control (see Chapter 3). Performance is improved by eliminating adverse control coupling, high autothrottle activity in turbulence, and by providing decoupled flight path and speed manoeuvre control that minimize elevator and throttle activity at each flight condition.

Envelope protection

Another problem associated with the SISO approach is that, since it is implemented to control speed through the throttle only, speed protection only works when the commanded flight path is within the aircraft's performance limits. For example, when thrust reaches its maximum limit due to excessive climb commands, the speed is left unprotected by the throttle–to–speed controller and speed may drop below stall speed if the path is not corrected. In most cases, when throttle reaches its maximum limit it is desirable to maintain commanded speed, which can then only be achieved through control of flight path.

The energy based MIMO controller inherently facilitates properly protecting speed. TECS protects the aircraft from over–speed by giving priority to speed control without intervention by separate speed protection systems. In other words, when TECS finds that the throttle control is inadequate for speed control (i.e., due to a maximum throttle limit) it uses the elevator to change the vertical flight path to satisfy the speed command. And similar to all other modes, intervention is achieved without switching inner–loop controllers, thus eliminating the undesired transient behavior that can occur with conventional designs.

System architecture

Over time, new modes and design provisions have been added to the flight deck. Often, these design provisions were added as patchwork to hide or compensate for the deficiencies described above. However, each provision proved to come with its own drawbacks, they degraded performance in other areas like control in turbulence, response to wind shear, and proper thrust response to intentional flight path changes. Fixing these deficiencies in turn resulted in a proliferation of control modes and an exceedingly complex system architecture from a point of view of design, maintenance, and operation by the flight crew (Lambregts, 1996).

These deficiencies cannot be fully overcome by addressing their symptoms at the man–machine interface level in the current legacy architectures. To effectively address all known deficiencies requires functional integration of all the subsystems, including the flight management system (FMS) and fly–by–wire (FBW) manual guidance and control functions. The MIMO controller serves as a generic elevator and thrust command computation algorithm that is capable of serving all flight path and speed control modes, providing consistency in behavior. Additionally, hardware and software are reduced significantly according to Lambregts (1996).

4.2.2 TECS evaluation

TECS was successfully flight tested on the NASA Langley Transport System Research Vehicle (TSRV), which is a highly modified Boeing 737-100 aircraft. Lambregts (1983b) gives a detailed overview of TECS performance during all modes and various flight conditions as the result of flight trials in 1984.

A complete report on the NASA flight trials performed in 1985 on the same aircraft is given by Bruce (1987). Pilots that flew with TECS reacted favorably, mainly due to the performance increase of the full time autothrottle control, the predictability and consistency of behavior in flight in all control modes, and throughout flight conditions, and the great precision that the Control Wheel Steering (CWS) mode allowed while greatly reducing the pilot's workload.

The experimental evaluations showed that TECS satisfied the stated design objectives. In addition, they supported Lambregts' claims that his design approach led to significantly better system performance. The latter allows us to find support for WDA as a generalized approach leading to better system performance and decreased system complexity.

4.2.3 Basic TECS architecture

Addressing the above challenges and design goals resulted in the conceptual TECS architecture shown in Figure 4.2, with the content of the 'TECS core' box detailed in Figure 4.3. The complete design and the implementation is much more complex and cannot be captured in a single diagram or in the scope of this chapter. For example, to achieve proper control law performance it is necessary to dynamically blend data from a variety of air–data and other sensors to produce synthesized feedback signals. This is a critical part of the design of TECS but it is left out of our discussions below without impacting, however, the principles illustrated.

TECS is divided in three main components: the *mode hierarchy*, which is the part upstream of the TECS core, and the TECS core that is divided into two parts: the *aircraft independent* part and the *aircraft tailored* part.

The mode hierarchy

The mode hierarchy provides the pilots with vertical path and speed control modes similar to the classical flight guidance and control systems, including maximum and minimum speed protection, flare, go around and control

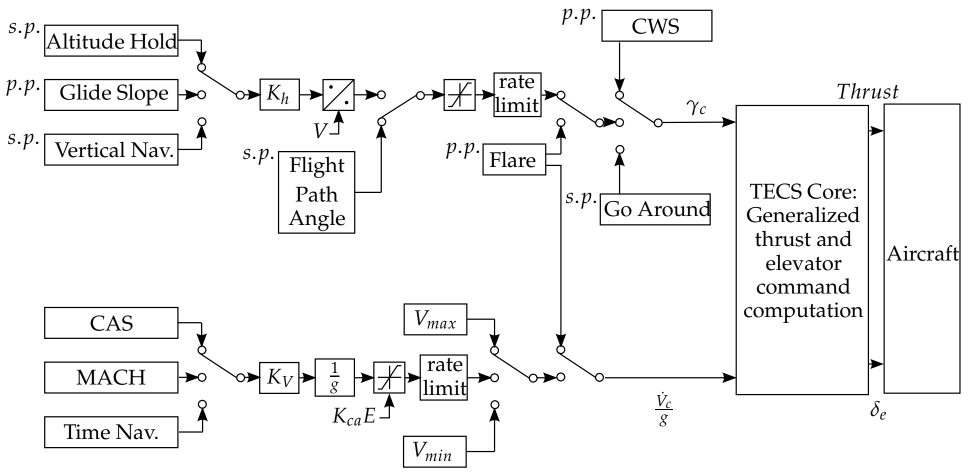


Figure 4.2: Basic TECS architecture and mode hierarchy including the TECS core as single block, adapted from Lambregts (1983a, 1983b).

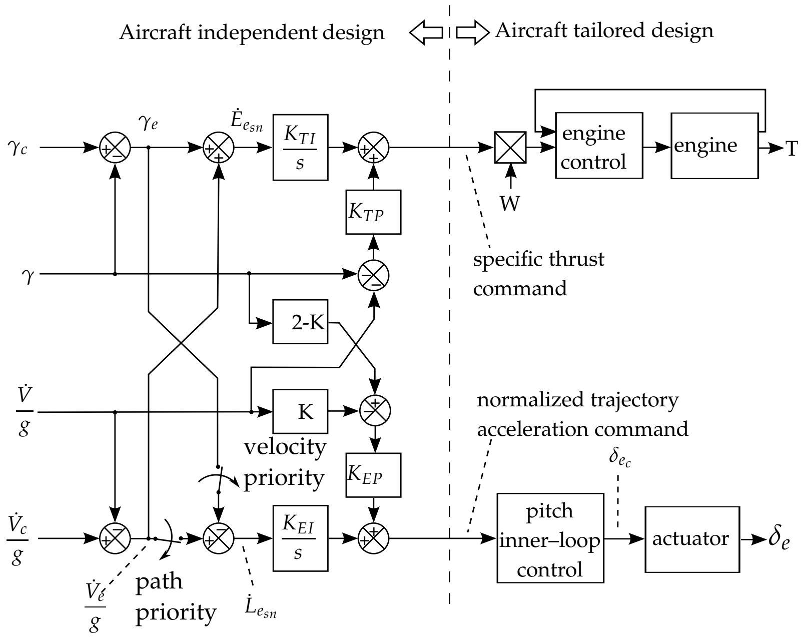


Figure 4.3: The aircraft–independent part of the TECS core (left) providing control decoupling, and the aircraft–tailored part of the TECS core (right) providing engine and elevator innerloop control. Adapted from Lambregts (1983a, 1983b)

wheel steering (CWS, manual FBW control). The signals produced by all modes represent deviations from the commanded vertical flight path and speed. These are further processed to become a vertical flight path angle command (γ_c) and a normalized acceleration command ($\frac{\dot{V}_c}{g}$) that the core processes. Duplication of functionality is avoided by designing a uniform core that is able to process commands from all possible modes.

The aircraft–independent part of the core

The aircraft–independent part of the core (Figure 4.3) provides the speed and altitude control decoupling, transforming the commands from the mode hierarchy into generic 'specific thrust' (pre–throttle) and 'normalized trajectory acceleration' (pre–elevator) commands. Except for the gain values that differ from aircraft to aircraft, this part is aircraft independent.

The aircraft–tailored part of the core

The aircraft–tailored part of the core (Figure 4.3) involves inner–loop pitch control and damping, engine control. The aircraft independent commands are turned into aircraft (and engine) specific throttle and elevator commands.

In the next section, these parts of TECS are further detailed using WDA, linking the functions of the components in the control diagrams to the goals of the complete system.

4.3 Work domain analysis, first chunking of TECS

Explanation of TECS and the WDA go hand in hand. In this section the three main parts of TECS are explained and at the same time the abstraction hierarchy framework is used to structure the knowledge. A *first chunking* of TECS is made as a first attempt at structuring the knowledge available on TECS, in terms of abstraction and part–whole aggregation. In this way, the reader should get an appreciation of the workings of TECS, but also an understanding of the intellectual process of mapping an automated system onto the abstraction hierarchy.

4.3.1 Approach

The system boundary is chosen around the aircraft, the air it flies in, and one additional Air Traffic Control (ATC) requirement that impacts flight control as we will see in section 4.3.6. It includes, for example, the aircraft, its dynamics, the control hardware, and its functioning, TECS, the TECS control panel that is operated by the pilots, and also turbulence and wind shear. It does not include other aircraft, the ATC system (for the one exception mentioned above), or specifics of the environment that the aircraft flies in.

Figure 4.4 shows the first chunking of TECS. Rasmussen's (1986) top three levels of abstraction are adopted: *functional purpose*, *abstract function* and *generalized function*. Abstraction relations are shown by the dashed solid head arrows, and aggregation relations are shown with the dashed hollow head arrows. Aggregation relations (opposite of part–whole decomposition) are used to link functions on the same level of abstraction in order to expand functions into sub–functions and features.

Functional purpose level

At the functional purpose level, 'TECS' denotes the entire system as a single concept. It is decomposed into three functional purposes: 'safety', 'production', and 'efficiency' according to van Paassen (1995). These provide a way to categorize the design requirements. Safety is mainly achieved by envelope protection e.g., protecting against over–speed and stall. Efficiency is decomposed into two goals achieved by TECS: reduction of fuel usage and lower strain on the engines, reducing maintenance cost.

Production is decomposed into a number of goals directly related to the primary function of TECS. Note that two production goals seem alike. One is 'speed and path control' and one is 'provide path and speed control modes'. The first represents the automated control over speed and path. The second represents the speed and path modes that the TECS system provides to the pilot as a means to control the aircraft. As shown, 'provide path and speed control' is achieved by the mode logic of the control mode hierarchy.

Abstract function level

The abstract function level represents the *organizing principles* that explain the reason (the why) for the system at the generalized function level to achieve the functional purposes. 'Energy based control decoupling' is the

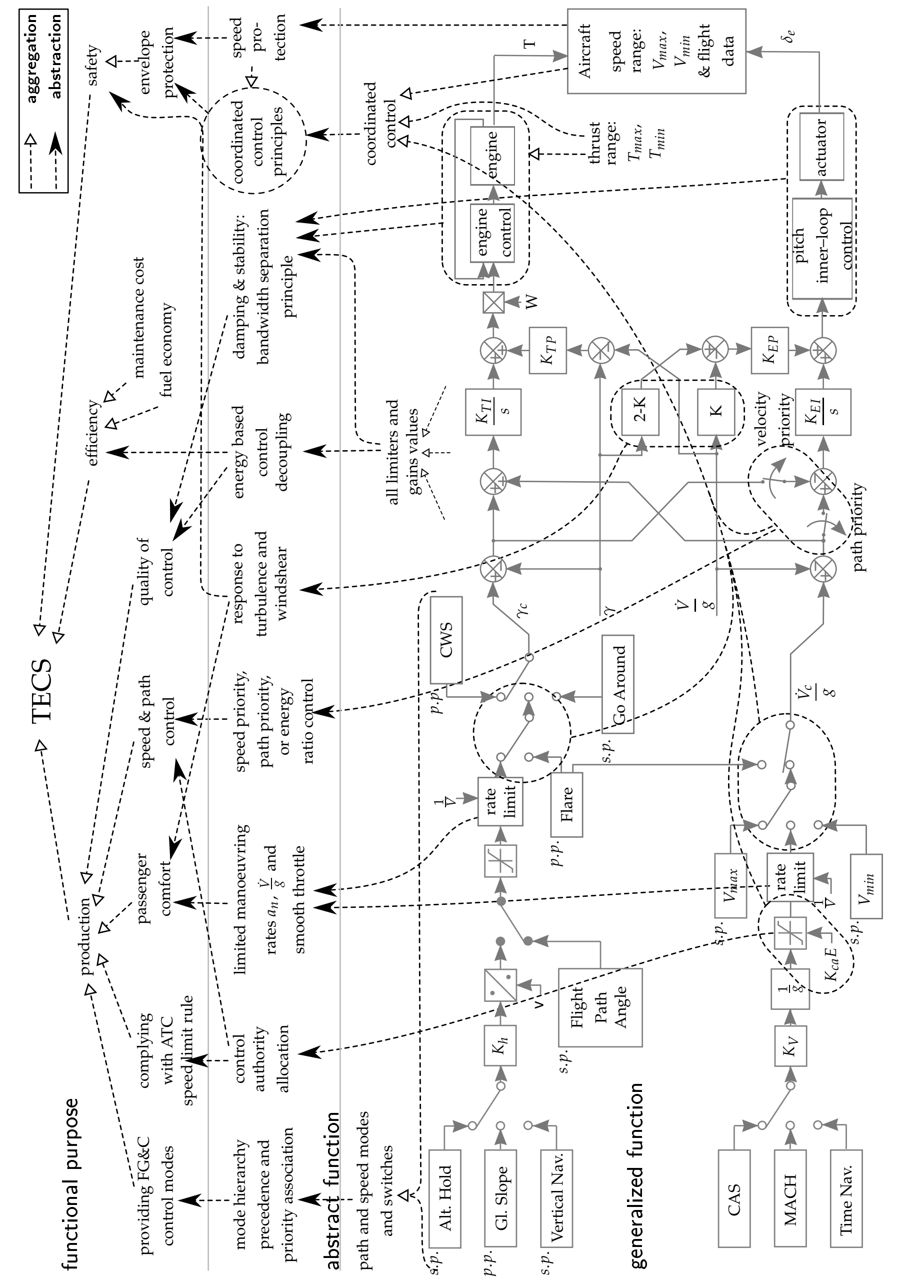


Figure 4.4: TECS mapped onto the abstraction hierarch showing how the components in the TECS control diagram on the generalized function level, achieve abstract functions and the functional purpose.

main organizing principle that Lambregts identified prior to design though the analysis of the fundamental physics of flight. However, this level represents many more design principles.

Generalized function level

The TECS control diagrams, Figures 4.2 & 4.3, are shown. They are conceptual and independent of physical implementation, and therefore belong to the generalized function level. These diagrams are the most concrete descriptions of TECS available.

Attempting to fill in the *physical function* and *physical form* levels without knowledge of the implementation will result in generic terms referring the airframe and control hardware without deepening insight in the workings of TECS. Therefore, the analysis here is limited to the top three levels of the abstraction hierarchy.

4.3.2 Aggregation

In this WDA the part–whole decomposition (aggregation) is not done in explicit levels as found in the abstraction decomposition space (ADS). As argued in Chapter 1, the nature of the part–whole decomposition relationship is determined by the level of abstraction at which it is. For example, at the generalized function level decomposition is component–oriented, while on the abstract function and functional purpose levels decomposition is function–oriented. Therefore, since a generalized set of part–whole decomposition levels does not fit all levels of abstraction this is not attempted, and the decomposition relation is shown only per level of abstraction.

Below, the three main parts of TECS are discussed in more detail using this framework defined above. The discussion will reference Figures 4.2 and 4.3 to explain signal processing functions and Figure 4.4 to explain what the signal processing functions *achieve* in relation to the purpose of TECS. Starting with the components found in the TECS control diagrams, Figures 4.2 and 4.3, on the generalized function level, the descriptions given are typically bottom–up.

4.3.3 Aircraft independent part of the TECS core

The aircraft independent part of the TECS core provides path and speed control decoupling. Inputs to the core are the vertical flight path command (γ_c) and the normalized acceleration command ($\frac{\dot{V}_c}{g}$). The outputs are the specific thrust command and the normalized trajectory acceleration command. These are further processed by the aircraft tailored part of the core.

Control decoupling is based on the principle that the throttle mainly controls the total energy rate of the aircraft and that the elevator, being energy conservative, controls the energy distribution rate between kinetic and potential energy. This was already discussed in Chapter 3, in Section 3.3.2 and was illustrated with the reservoir analogy of Figure 3.1.

The TECS core inputs can be read as energy rates relative to the air mass. The total energy of the aircraft is given by:

$$E = mgh + \frac{1}{2}mV^2 \,,$$

which can be differentiated to give the total energy rate $\dot{E}$:

$$\dot{E} = mg(\dot{h} + V\frac{\dot{V}}{g}) \,. \tag{4.1}$$

By division by mg the specific total energy rate $\dot{E}_s$ is given:

$$\dot{E}_s = V(\gamma + \frac{\dot{V}}{g}) \,, \tag{4.2}$$

and finally through dividing by the true airspeed V, the specific non–dimensional energy rate is given:

$$\dot{E}_{sn} = \gamma + \frac{\dot{V}}{g} \,. \tag{4.3}$$

Here, γ represents the specific non–dimensional potential energy rate, and $\frac{\dot{V}}{g}$ represents the specific non–dimensional kinetic energy rate, which are inputs for the TECS core. In combination with the commands from the mode hierarchy, the specific non–dimensional potential energy rate error (which equals the vertical flight path angle error), $\gamma_\epsilon = \gamma_c - \gamma$, and the specific non–dimensional kinetic energy rate error (which equals the normalized acceleration error), $\frac{\dot{V}_\epsilon}{g} = \frac{\dot{V}_c}{g} - \frac{\dot{V}}{g}$, are computed. These values are used to calculate the specific non–dimensional total energy rate error:

$$\dot{E}_{\epsilon_{sn}} = \gamma_\epsilon + \frac{\dot{V}_\epsilon}{g}, \tag{4.4}$$

and the specific non–dimensional energy distribution rate error:

$$\dot{L}_{\epsilon_{sn}} = \frac{\dot{V}_\epsilon}{g} - \gamma_\epsilon. \tag{4.5}$$

The energy rate error is used to compute the normalized specific thrust command:

$$\delta T_c = -K_{TP}\dot{E}_{sn} + K_{TI}E_{\epsilon_{sn}}. \tag{4.6}$$

The energy distribution rate error is used to compute the normalized trajectory acceleration command:

$$nta_c = -K_{EP}\dot{L}_{sn} + K_{EI}L_{\epsilon_{sn}}, \tag{4.7}$$

from which the elevator command δ_{ϵ_c} is computed by the inner–loop elevator control. To preserve the energy relationship and ensure identical dynamics for altitude and speed control, the gains in the core are chosen to be (Lambregts, 1983b):

$$K_{TP} = K_{EP} = 1.0 \text{ and,} \tag{4.8}$$

$$K_{TI} = K_{EI}. \tag{4.9}$$

The value of these latter two integrator gains depends on the aircraft inner–loop dynamics as will be discussed below.

Energy based control decoupling

All gains in the TECS control diagram are chosen such that the dynamics in speed, path, throttle and elevator control are the same to preserve the energy relationship and achieve the *energy based control decoupling*. Figure 4.4 shows this by linking the 'limits and gains values' on the generalized function level to 'energy based control decoupling' on the abstract function with the abstraction arrow.

Bandwidth separation principle

At the same time the values of the gains need to satisfy the bandwidth separation principle to ensure damping and stability. This is a control engineering principle for achieving stability and damping in a system that consists of various nested control loops. The control bandwidth of the different loops (which are: engine thrust and pitch attitude, flight path angle and longitudinal acceleration, speed and altitude) are selected to be a factor of 3.3 to 7.5 apart (Lambregts, 1996). This is achieved by proper gain selection in the control loops. In TECS, the application of this principle achieves damping and stability, thus an arrow is drawn from 'limits and gains values' to 'damping & stability: bandwidth separation principle'.

Quality of control

Together, the 'energy based control decoupling' and 'bandwidth separation' organizing principles on the abstract function level, achieve 'quality of control' on the functional purpose level, addressing the problems with conventional systems as discussed in Section 4.2.1. 'Energy based control decoupling' also achieves efficiency in terms of fuel economy and maintenance cost due to reduced engine wear.

The γ and $\frac{\dot{V}}{g}$ signals are affected by turbulence and wind shear. Lambregts introduced the gain K in the blocks K and $2 - K$, found in the proportional control paths in the core, Figure 4.2, to be able to achieve the desired trade–off between speed and path deviations through selecting value for K between 0 and 2. In Figure 4.4 the abstraction arrow from the K and $2 - K$ blocks to 'response to turbulence and windshear' illustrates this function. The response to turbulence and windshear is selected such that it achieves the purpose of passenger comfort (mainly turbulence) and safety (mainly windshear). According to Lambregts (2009), $K = 1$ was chosen, resulting in unity gains, for best overall performance.

Speed and path control

The TECS core can compute the elevator and throttle commands in three different ways, depending on the positions of the cross-feed switches in Figure 4.3. First, with both switches closed, the default configuration is active. The normalized acceleration and path rate errors are nulled equally, giving equal priority to speed and path. With the thrust command at its limit (i.e., the engines are at maximum or minimum thrust) one of the commanded variables (speed or path) cannot be satisfied. Priority then has to be given to either speed or path.

Second, speed priority can be achieved by opening the speed priority cross-feed switch in the core, which disables the γ_ϵ feedback in the elevator integrator. Integral elevator control is then based on the $\frac{\dot{V}_\epsilon}{g}$ signal only, resulting in a speed–to–elevator control strategy.

Third, path priority can be achieved by opening the path priority cross-feed switch in the core that disables $\frac{\dot{V}_\epsilon}{g}$ feedback in the elevator control integrator (Figure 4.3). Integral elevator control is then based on the γ_ϵ signal only, resulting is a path–to–elevator control strategy.

These three modes are basically organizing principles for solving the flight control equation under different circumstances, and as such they are represented at the abstract function level as'speed priority, path priority or energy ratio control'. 'Speed and path control', shown as part of the production goal of TECS on the functional purpose level, is achieved by the selected mode on the abstract function level. In turn, this abstract function is achieved by setting the mode priority switches shown at the generalized function level, Figure 4.4.

Each vertical flight path mode has speed priority or path priority associated with it, as denoted in Figure 4.2 with 's.p.' and 'p.p.', respectively, next to the path modes. The speed modes do not have a priority configuration associated with them because they are always engaged in combination with a path mode that has either a path or a speed priority associated with it. In either priority mode, when speed approaches V_{max} or V_{min}, the appropriate speed protection mode (see Figure 4.2) takes over and speed priority is set by 'coordination control logic'.

The above explains how the components and their configuration in the aircraft independent part of the TECS core, achieve a number of the design goals of TECS. The components and their role in TECS have been discussed in reasonable detail to show how the design considerations at the generalized function level satisfy the guiding principles at the abstract function level, and eventually at the functional purpose level.

The TECS core is discussed first because it contains the energy based control decoupling, which is the main design principle of TECS. Below the discussion continues on the aircraft tailored part of the TECS core, and then for the TECS mode hierarchy.

4.3.4 Aircraft tailored part of the TECS core

The aircraft tailored part of the TECS core addresses the inner–most control of TECS. It is aircraft tailored because the inner–loop elevator control is designed to control the aircraft's pitch response, and the engine control is designed around the engine dynamics. This part of the core controls the throttle and the elevator based on the specific net thrust command and normalized trajectory acceleration command that are generated by the aircraft independent part of the TECS core.

In the engine control path, the specific net thrust command is multiplied by the aircraft weight (W) to yield the net thrust command. The net thrust command is divided by the number of operational engines and fed to each individual engine control. This effectively turns the engines into net thrust actuators.

The elevator inner–loop control adds pitch damping terms to the normalized trajectory acceleration command to yield the elevator command. Pitch and thrust responses must be matched to produce decoupled and coordinated control. The engines must match the net thrust command at all flight conditions in order to achieve true command decoupling. Engines typically have the slowest dynamics, and therefore the pitch inner–loop elevator control dynamics are matched to the thrust dynamics by a judicious choice of the inner–loop feedbacks and gains (Lambregts, 1996). The variable elevator effectiveness is compensated by gain scheduling.

The bandwidth separation principle is applied to determine K_{TI}, K_{EI}, K_{TP}, K_{EP} in the TECS core and K_V and K_h in the mode hierarchy based on the engine dynamics. In Figure 4.4 the gains are part of 'limits and gains values' and are set to achieve the 'bandwidth separation principle'.

4.3.5 The TECS mode hierarchy

The TECS core, as discussed in the two parts above, provides the energy based control decoupling and control of throttle and elevator. The TECS mode hierarchy provides control modes that generate the vertical flight path command (γ_c) and the normalized acceleration command ($\frac{\dot{V}_c}{g}$) to the TECS core. Below, the discussion continues on the components of the mode hierarchy on the *generalized function level*, and how these achieve functions at the abstract function level and at the functional purpose level.

Mode precedence

The modes are organized in a hierarchy to achieve precedence of certain modes over other modes, as shown in Figure 4.2. For example, to prevent V_{min} protection during the flare maneuver, the Flare mode has precedence over the V_{min} and V_{max} speed protection due to the configuration of the logical mode switches. The speed protection modes, in turn, have precedence over the other speed control modes, to prevent stall and excessive speeds. A similar hierarchical structure is found for the path modes.

Path modes

The Altitude Hold, Glide Slope and Vertical Nav. modes generate an altitude error signal relative to the commanded vertical flight path. The altitude error is multiplied by the gain K_h to produce the vertical speed command $\dot{h}_c$:

$$\dot{h}_c = K_h h_\epsilon, \tag{4.10}$$

which is then divided by the true airspeed V to yield the vertical flight path angle command γ_c:

$$\gamma_c = \frac{\dot{h}_c}{V}. \tag{4.11}$$

The rate limiter downstream of the modes limits the commanded manoeuvring rates and will be further discussed below. The Flight Path Angle mode produces a vertical flight path angle command internally. The CWS, Flare, and Go Around modes also produce a flight path angle command internally but are not limited by the rate limiter to enable full manoeuvring capability in these modes. The amplitude limiter downstream of the Flight Path Angle mode is not used in the final design of TECS according to Lambregts (Lambregts, 2009). The maximum and minimum commanded flight path angles are inherently limited by the aircraft and engine performance characteristics.

Speed modes

The CAS (calibrated air speed), MACH, and Time Nav. speed control modes generate the true airspeed error that is multiplied by gain K_V and divided by the gravitational constant g to yield the commanded normalized acceleration along the flight path $\frac{\dot{V}_c}{g}$:

$$\frac{\dot{V}_c}{g} = \frac{K_V V_\epsilon}{g}. \tag{4.12}$$

Downstream of these modes are the normalized acceleration command amplitude limiter and the rate limiter, similar to those applied in the path command signal path. The amplitude limiter is used only when executing simultaneous flight path and speed command manoeuvres when thrust is at the limit in speed priority mode. This is called *control authority allocation*, which is shown at the abstract function level in Figure 4.4, and further discussed below. The rate limiter limits the commanded manoeuvring rates and is discussed in the following.
The magnitudes of the gains K_V and K_h are chosen equal to yield identical dynamics for speed and altitude manoeuvres. The value is selected to provide the desired response bandwidth consistent with the inner–loop gains, such that the responses are overshoot free, thus critically damped. This is represented in the abstraction hierarchy by 'limits and gains values' on the generalized function level, achieving 'energy based control decoupling' and 'bandwidth separation principle' on the abstract function level.

Rate limiters

Both rate limiters constrain the rate of change of the commanded energy state and provide smooth speed and path command transitions. They are a means to limit the normal acceleration, a_n, and rate of change of speed, $\dot{V}$, mainly for passenger comfort. As shown in Figure 4.4, the 'rate limits' achieve 'limited maneuvering rates' as abstract function that achieves 'passenger comfort' at the functional purpose level.
The rate limiters need to have equal values to yield identical dynamics for speed and altitude to achieve the 'energy based control decoupling'. The value is derived from the maximum allowable normal acceleration $a_{n_{LIMIT}}$, in this case required for passenger comfort. The normal acceleration is given by $a_n = V\dot{\gamma}$, thus the rate of the γ_c signal is limited to:

$$\left(\frac{d}{dt}(\gamma_c)\right)_{LIMIT} = \frac{a_{n_{LIMIT}}}{V}. \tag{4.13}$$

The same rate limiter is applied to the $\frac{\dot{V}_c}{g}$ signal:

$$\left(\frac{d}{dt}\left(\frac{\dot{V}_c}{g}\right)\right)_{LIMIT} = \frac{a_{n_{LIMIT}}}{V}. \tag{4.14}$$

The speed protection mode (V_{max} and V_{min}) and the Flare mode generate normalized acceleration commands internally. These modes are not constrained by the rate and amplitude limiters because they must be allowed maximum safe vertical manoeuvering.

4.3.6 Control authority allocation

In Figure 4.4 an abstraction arrow is drawn from the amplitude limiter in the speed signal path to the principle of 'control authority allocation' on the abstract function level. In turn 'control authority' achieves 'speed & path control' and 'comply with ATC speed limit rule' on the functional purpose level.

Control authority allocation is a function implemented to deal with the situation where the thrust command is at its limit and *simultaneous* path and speed commands are given. This is typically a situation where control takes place at the *edge* of a system where the aircraft specific constraints need to be taken into account (van Paassen & Mulder, 2004).

In this situation only elevator control is left to parse the available total energy rate between speed and path commands. In most cases speed priority logic is used and all of the total manoeuvre authority ($\dot{E} = \gamma + \frac{\dot{V}}{g}$) would be allocated to execute the speed manoeuvre first, and then to execute the path manoeuvre command. Control authority allocation can allocate some of the total manoeuvre authority to path control as the speed manoeuvre is executed in speed priority mode.

At maximum thrust command, $\dot{E}$ represents the maximum achievable energy rate, and at minimum thrust $\dot{E}$ represents the minimum achievable energy rate (depending on the drag configuration, e.g., flaps, landing gear).

The amplitude limiter in the speed command signal path in Figure 4.2 is only used in this situation to limit the amplitude of the normalized acceleration command ($\frac{\dot{V}_c}{g}$) to the value $K_{ca}E$, where K_{ca} can be selected between zero and unity. With $K_{ca} = 0$, full manoeuvre authority is allocated for path control. With $K_{ca} = 1$, full manoeuvre authority is allocated for speed control.

For commanded climbs, $K_{ca} = 0.5$ was selected so that 50% of the manoeuvre authority is allocated for path commands and 50% is allocated for acceleration commands. This results in maximum performance climbs with temporary reduction of 50% in path angle until the speed command change has been executed. Then 100% of the maximum energy rate is dedicated to the path command.

On descents, 100% of the manoeuvre authority is allocated for deceleration commands by setting $K_{ca} = 1.0$. The motivation behind this setting is to allocate full manoeuvre authority to speed control to comply with the 250 kts ATC speed limit rule below 10,000 feet. A large deceleration command will temporarily reduce the descent rate to zero until the commanded speed is captured. A full overview of the control authority logic as well as the speed/path priority logic is given by Lambregts (1996).

4.3.7 Coordination logic

The control authority allocation illustrates the need for coordination logic that: detects and distinguishes certain situations (e.g., thrust at its limit) and coordinates mode switches (e.g., speed protection), sets the speed/path priority logic according to active path mode, engages the amplitude limiter, sets the value for K_{ca} and more. Available literature does not describe the internal functioning of this logic, neither does it describe the internal logic of each of the modes in much detail. It can be assumed that the undisclosed logic follows the same principles described so far that ensure the energy relation between the path and acceleration signal paths, and avoid duplication of functions throughout the system. It is also assumed that the logic is consistent with Lambregts' design goal to eliminate unnecessary complexity, in line with the argument that Lambregts (1996) makes for the need for TECS.

The coordination logic is represented on the generalized function level in Figure 4.4 and numerous aggregation arrows are drawn to it from components that are part of the coordination logic without being able to represent the internals of the control logic itself. The aircraft safe speed range and the engine's thrust range are shown to be part of the 'coordinated control' at the generalized function level, to illustrate that these values are used by the logic to detect when thrust is at its limit, configure switches, set the control authority, etc. The 'coordinated control principles' are linked with many functions in TECS, to prevent clutter in the figure these links are not drawn.

4.4 Further work domain analysis

Figure 4.4 shows a first chunking of TECS, which is a mapping of available, and selected, knowledge of TECS onto the abstraction hierarchy. This is a first step in WDA taken to analyze an automated system. It differs from the approach in Chapter 3 because the automation is part of the analyzed system. This section shows how an existing system can be mapped onto the abstraction hierarchy, which will be useful when performing a WDA for an evolving system during the design process, that will be the subject of Chapter 5, for a mini UAV system.

4.4.1 Aggregation and abstraction

The first chunking of TECS maps out knowledge of the system with respect to how the system components are designed to achieve the goals of TECS. By making abstractions and aggregations, the analysis focuses on the essential abstract principles underlying the functioning of the system and therefore the design choices.

By viewing Figures 4.2 and 4.3, it is hard to see how the energy relations are instantiated, although the cross-feeds in Figure 4.3 do provide a hint. At this level of part–whole decomposition, the system is viewed in terms of signal processing, gain values, and location of the amplitude limiters, integrators, etc. A natural abstraction from this representation and level of part–whole decomposition could be toward control–theoretical analysis, covering control action response, transient response, stability, frequency response and robustness. However, from our perspective, energy is the dominant constraint around which the design is organized. It is our goal to make abstractions towards the energy representations, therefore, the level of part–whole decomposition needs to change.

Figure 4.5 shows an aggregation where the components of Figures 4.2 and 4.3 have been grouped in such a way that the aircraft–independent part of the TECS core becomes the main focus. The arrows in Figure 4.5 still represent signals but a lot of detail (rate limiters, gains, etc.) has been removed, including the control theoretical considerations. On this level of part–whole decomposition, TECS can be seen in its three main parts. First, the mode hierarchy that provides means to control path and speed; second, the generic energy based core that provides control decoupling, and third, the aircraft specific parts that provide controlled flight. These three parts can be visual-

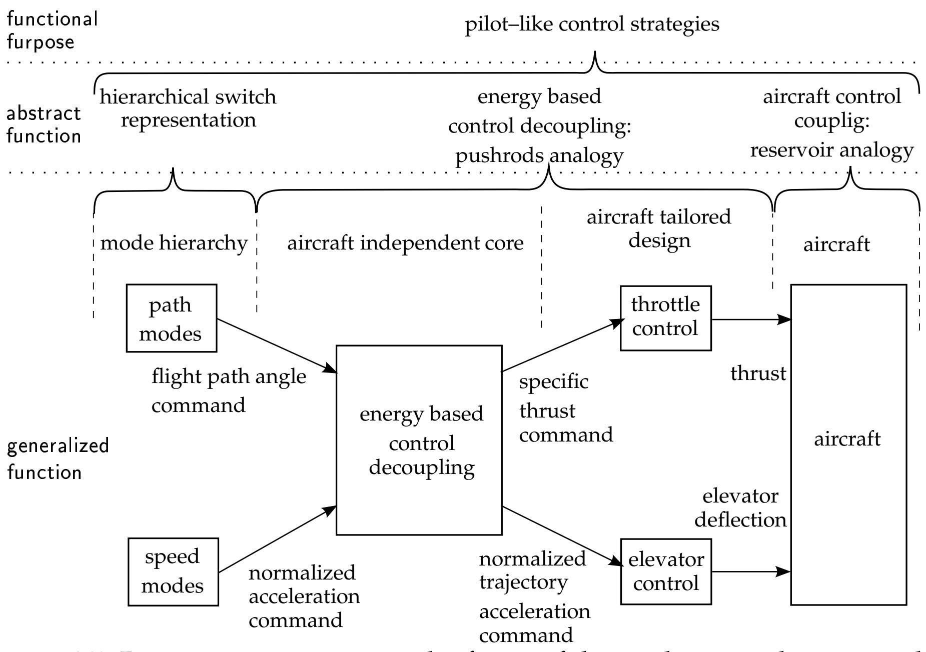

functional furpose
pilot–like control strategies
abstract function
hierarchical switch representation
energy based control decoupling: pushrods analogy
aircraft control couplig: reservoir analogy
mode hierarchy
aircraft independent core
aircraft tailored design
aircraft
generalized function
path modes
flight path angle command
energy based control decoupling
specific thrust command
throttle control
thrust
aircraft
elevator deflection
speed modes
normalized acceleration command
normalized trajectory acceleration command
elevator control

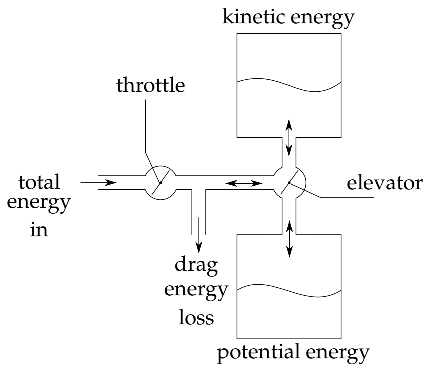
Figure 4.5: By using aggregations, the focus of the analysis can be put on the energy control strategies of TECS.

ized on the abstract function level with an analogy to illustrate the abstract functions.

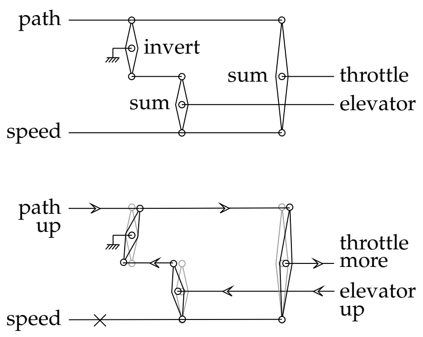
Figure 4.6(a) again shows the aircraft reservoir analogy, as introduced in Section 3.3.3. The aircraft is shown as a system storing two kinds of energy in two reservoirs: kinetic energy (speed) and potential energy (height). The throttle controls the total energy inflow, whereas the elevator controls the distribution of the total energy flow between the two energy reservoirs. The problem of control decoupling is immediately evident: when the (auto-) pilot wants to meet speed and/or altitude goals, a coordination of throttle and elevator is needed because neither the throttle nor the elevator controls speed or vertical path alone.

The TECS aircraft–independent core is designed to translate path and speed commands into total energy and energy distribution commands to match the aircraft energy controls. The fact that the controls should be coordinated is an important insight and suggests that the SISO control strategy of separate autopilot (elevator) and autothrottle systems is indeed a poor choice.

To match the reservoir analogy representing the aircraft control coupling,



(a) Reservoirs analogy representing the aircraft control coupling in terms of energy. The arrows represent energy flows and the throttle and elevator act as valves to control them.



(b) Pushrods analogy representing TECS control decoupling in terms of a mechanical representation of the control algorithm. The arrows show pushrods movements and an $\times$ indicates no movement.

Figure 4.6: Analogies to describe (a) the inherent aircraft energy coupling and (b) the control decoupling by TECS.

an analogy can be made to represent the control decoupling that TECS provides. This analogy is introduced as the pushrods analogy, shown in Figure 4.6(b). It is a mechanical representation of the simplified mathematical relations designed into the aircraft–independent core. The top diagram in Figure 4.6(b) shows the analogy in rest, while the lower diagram exemplifies the case with control inputs. One can mentally experiment with the speed and path control inputs to see which throttle and elevator commands are produced. For example, as illustrated in 4.6(b), a positive path input (up), pushing the path rod to the right, will cause a positive input on the throttle, and a negative input on the elevator. The outputs can be mentally linked to the inputs of the reservoir analogy, in Figure 4.6(a), to see how the aircraft controls qualitatively direct the total energy flow into the potential energy reservoir. Continuing this example, the throttle control provides the total energy flowing in, and elevator control achieves that the energy flows into the potential energy reservoir, representing an increase in altitude, while keeping the kinetic energy level, thus speed, constant.

The different core control actions can be visualized with the energy matrix, introduced in Chapter 3 (Amelink et al., 2005b), as shown in Figure 4.7. The top–left to bottom–right diagonal shows situations where the total energy

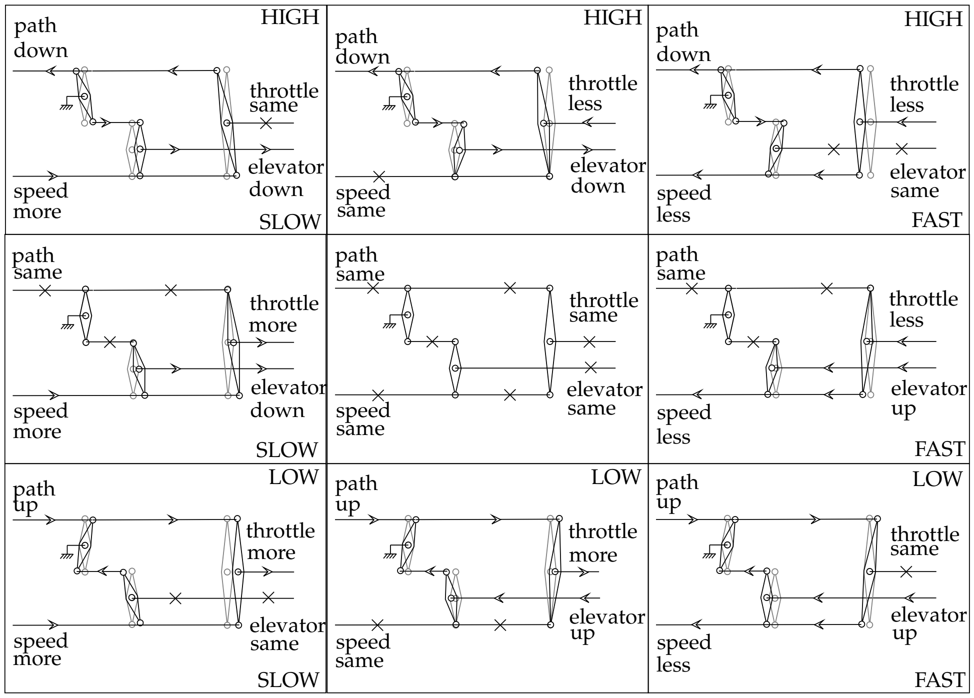
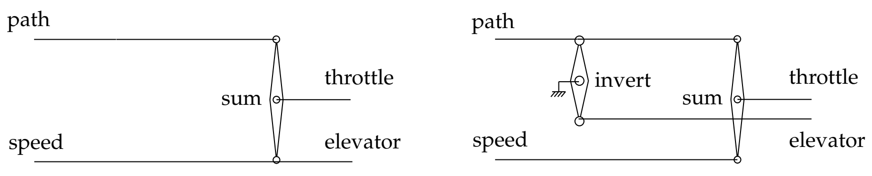


Figure 4.7: The energy state matrix used with the pushrods analogy, translates speed and altitude deviation into coordinated throttle and elevator commands.

error is null and speed and altitude deviations can be canceled out with only elevator control. The top–right to bottom–left shows situations with a total energy error but where the energy ratio error is null. The TECS core will initially respond with throttle control to eliminate the total energy error, and then use the elevator to keep the energy ratio error null.

Besides the default core configuration to minimize energy ratio errors, the core can operate in speed priority and path priority modes. The priority modes provide different solutions to the flight control problem leaving either path or speed the uncontrolled variable, respectively, in the situation when thrust is at its limit. Figure 4.8 shows the equivalent pushrods analogy for speed and priority path modes, illustrating that the controller strategies are indeed different from the default control strategy.

The hierarchical relationship between the modes can best be visualized by the switching logic already shown at the generalized function level, but



(a) Speed priority logic (b) Path priority logic

Figure 4.8: Pushrods analogies for the speed and path priority modes, showing that these priority modes solve the control problem differently than the default energy ratio control of TECS.

without the gains and limits to focus on the hierarchical relation between the modes. Each mode represents a configuration of TECS to achieve the path and speed goals in a pilot selected manner.

4.4.2 More than energy control

In Chapter 3 the system boundary was chosen to constrain a single control problem: energy control. This was done to be able to have a clear control problem in the abstraction hierarchy and not mix constraints on, e.g., air traffic control or the transportation system with the vehicle control problem (Amelink, 2002). TECS, however, as described in the previous sections, spans at least three control problems and the same simplification cannot be made.

As shown at the beginning of this chapter, the considered part of TECS can be divided into three main parts: path and speed control modes, control decoupling, and inner–loop elevator and throttle control. In Figure 4.5 these parts are shown as aggregated blocks on the generalized function level. Although energy management is present in each part (through limits and gains values) each part basically deals with another part of the total control problem. Together they represent nested control functions that are not addressed by the abstraction hierarchy or ADS.

We used aggregation to chunk parts of the system together and put the focus on the energy relationships (Figure 4.5). This can be done for other principles as well, depending on the control problem of interest. We have identified that the nested structure of TECS lends itself well for separating the control problems.

Our intention is to capture their nested structure in WDA. The three parts of TECS, and their associated control problems are analyzed using the abstraction hierarchy. The resulting abstraction hierarchies are stacked in a second hierarchy of levels of *control sophistication*. This term reflects the achievement of higher order, more sophisticated control by adding layers of automation. Together, the abstraction hierarchies and the levels of control sophistication form the Abstraction–Sophistication Analysis (ASA) framework.

Figure 4.9 shows TECS mapped out on the ASA. Two more levels of control sophistication have been added: the *aircraft* as the bottom level, and the 'coordination logic' mentioned earlier as the top level that coordinates the levels below. The levels of abstraction are along the horizontal axis and the levels of control sophistication are along the vertical axis. Note that 'coordination logic' has been added as the top level of control sophistication because it has the coordinating function over all other levels. Below it are the three parts to TECS; 'flight control modes', 'ac independent control decoupling', and 'ac dependent control'. The lowest level is the 'aircraft' because it provides the flying platform on which control is implemented.

The abstraction–sophistication analysis

The ASA framework provides another view on the system. The column of functional purpose levels lists all top level functions in the system. At the base is the airliner jet, with energy conservative elevator assumption (see Section 3.3). Moving up in the column we find control functions rising in level of sophistication: generic energy control inputs, pilot–like control strategies, FG&C control modes and passenger comfort, and speed protection. The latter needs some further explanation: the mode hierarchy provides the V_{max} and V_{min} speed protection modes but the coordination logic provides the automatic selection of the modes as well as other switching logic that activates control authority allocation and speed/path priority modes to ensure envelope protection.

At the abstract function level, at each level of control sophistication the *command transformations* are represented. This is inspired by Marr's (1982) second level: *representation and algorithm* (discussed in Section 2.3).The principles of the command transformation are represented on the abstract function level, and the implementation in terms of signal processing paths and blocks is represented at the generalized function level. The command transformations and input and output representations link the abstract function

		← abstraction ←		
		functional purpose	abstract function	generalized function
control sophistication ↑	coordination logic	Safety: -setting V_{max} or V_{min} mode Other: -setting control authority allocation -setting priority mode	Configuration principles: -set speed, path priority -set Max/Min speed mode -control authority allocation	Coordination logic: -thrust limit detection -speed limit detection -configuration switches Control modes
↑	flight control modes	Provide FG&C automatic control modes to pilot. Passenger comfort.	Hierarchical mode structure. Principles of each speed and path mode. Path and normalized acceleration command generation. Limited manoeuvre rates. Bandwidth separation.	Mode Hierarchy: -speed and vertical -path modes -rate limiters. Acceleration, path control
	ac independent control decoupling	Pilot–like control strategies.	Energy based control decoupling: pushrods analogy. Command transformation: Norm. acceleration and path commands to specific thrust and norm. trajectory acceleration commands. Bandwidth separation.	Aircraft independent core diagram. Generic energy control
↑	ac dependent control	Generic energy control inputs.	Command transformation: specific net thrust to thrust, and normalized trajectory acceleration to elevator deflection. Matched elevator and engine dynamics.	Engines and engine control. Elevator and innerloop elevator control. Controllable flight
	aircraft	Airline jet	Aircraft as energy system: -reservoir analogy. Aircraft pitch dynamics.	Flight data: speed and path. Safe speed range. Engine thrust range.

Figure 4.9: TECS as discussed in this chapter mapped onto multiple abstraction hierarchies and levels of control sophistication, forming the Abstraction–Sophistication Analysis (ASA).

levels of Figure 4.9 vertically. Often it helps to visualize the algorithms with analogies such as the reservoir analogy and the pushrods analogy.

The generalized functions in the ASA correspond to the division and aggregation performed in Figure 4.5. At the *aircraft* level, the state variables are represented, similar to the cause–effect diagram in Figure 3.3. Note that the abstraction hierarchy of the control task analysis of Chapter 3 differs from the one presented in this chapter at the *aircraft* level. This is mainly due to evolving insight into mapping domain knowledge onto the abstraction hierarchy. Each additional level of control sophistication adds elements of the TECS control diagrams according to which level of control sophistication they belong.

The curly brackets in–between the levels of control sophistication show that each whole level of control sophistication provides a generalized function to the level above. It shows that the structure of the individual abstraction hierarchies is nested, and that each level of control sophistication is based on all levels below it. To illustrate: the FG&C modes at the *flight control modes* level can only function when the autopilot, autothrottle, elevator control, and all other functions on the levels below are functional as well.

More than constraints

The ASA shows not only the constraints that work on the control functions but also the control functions themselves. At the aircraft level, the same constraints are represented that work on the human control task (of Chapter 3). New constraints are introduced at higher levels of control sophistication when layers of automation are designed. These constraints shape higher levels of control and are to be taken into account at these higher levels. At the same time, the higher levels put requirements on the design of the lower levels.

Therefore, the ASA framework provides a way to work with automation introduced constraints in WDA, without having to look at the entire system in one representation. However, working with the ASA does not eliminate the need to look at the entire system, it merely helps to separate and analyze control problems inherent to the domain, vehicle, and desired level of control.

4.5 Discussion

This chapter discussed TECS by using the abstraction hierarchy as a means to structure the available knowledge on TECS. The approach to the design of TECS started with the analysis of the physics of flight that led to identification of the energy constraints on flight, and their explicit inclusion in the design of TECS. Energy management became a main organizing principle for the control system, which was unlike other flight control systems at that time. Due to the analysis preceding the design of TECS, it is considered an example of the ecological approach to automation design. The increased performance of TECS over classical designs and the claimed reduction in complexity indicate the potential value of this approach.

To generate insights into this approach, the available knowledge on TECS was mapped onto the abstraction hierarchy. The abstraction hierarchy itself does not provide answers to design challenges, but it provides a means to structure knowledge of the system and understand the reasons for its design to a detailed level. By working with the abstract function level, the designer is compelled to think about the system's organizing principles that govern system behavior. Different representations at the abstract function level enable different views on the system.

In Chapter 3, the aircraft was depicted using the reservoir analogy, at the abstract function level, to visualize the control *coupling* between elevator and throttle. The same representation was used in this chapter, and it is matched by the pushrods analogy to visualize the control *decoupling* designed into TECS. These analogies show that TECS is an example where the automation constraints, represented by the pushrods analogy, are aligned at the abstract function level with the system constraints, represented by the reservoir analogy.

The generalized function level captures the TECS control diagrams that is independent of actual implementation on the flight control computers. The control diagrams, Figures 4.2 and 4.3, contain many design considerations that are captured in the component layout and in the limiter and gain values. These are made explicit by linking them to their organizing principles that represent the reason for their inclusion in the design.

4.5.1 Generated insights

The available knowledge of TECS could be mapped onto the abstraction hierarchy, shown in Figure 4.4. The means–ends relationships across the levels of abstraction allowed the linking of parts of the control diagram at the generalized function level to the system's organizing principles at the abstract function level and then to the system's purpose. This way, the system purpose and abstract functions are shown as the reason for design choices that were made at the generalized function level. The means–ends structure serves to explain the design, as has been demonstrated in this chapter.

Figure 4.4 shows how parts of the control diagram at the generalized function level are linked to the more abstract levels. However, it is not just the components of the control diagram that provide the abstract functions but the structure interrelating them and their co–functioning. This is best illustrated by the aircraft independent part of the TECS core control diagram, see Figure 4.3. This entire structure, including all gain and limiter values, is responsible for the control behavior that achieves 'energy based control decoupling' at the abstract function level.

From literature (Lambregts, 1996) it is clear that there is an elaborate control function that coordinates control between all the levels exists in the system, that interacts with the mode selection switches, control authority and more. In fact TECS is much more complex than the available literature explains (Lambregts, 2009). Missing parts that are identified but not filled in can be part of the abstraction hierarchy. However, there is no evidence to assume that the abstraction hierarchy helps to identify all white spots. The abstraction hierarchy, as a structure to represent knowledge, allows the analyst to reason about the validity of the knowledge he is mapping. In fact, the process of mapping the knowledge onto the abstraction hierarchy leads the analyst to ask questions about the available knowledge and look deeper into it. However, the structure that the abstraction hierarchy provides does not provide a methodological check for validity or truth of the represented knowledge.

The abstraction hierarchy is not a specific modeling language and does not support specific means–ends relationships. For example, how the TECS control diagram achieves 'energy based control decoupling' is left to the analyst to understand based on his interpretation of the control diagram. Once this function is understood, it can be represented in the abstraction hierarchy. However, changes to the control diagram need re–interpretation of the new diagram to establish if it still achieves the 'energy based control decoupling'. Thus, human intelligence is needed to interpret the abstraction hierarchy representation of the system.

In Chapter 3, the abstract function level was dominated by the energy relationship that governs flight. By mapping the knowledge of TECS, the energy relationship is only one of the many functions at the abstract function level. Other functions govern the behavior of the control system as well. The need for the 'control authority' function to 'comply with the ATC speed rule' illustrates that artificial constraints that do not result from properties of the physical world can be introduced by the designer in the system to produce the desired behavior.

Parts of the control diagram are aggregated in larger blocks to focus the analysis on the 'energy based control decoupling', see Figure 4.5. Aggregation is used to select and focus on the energy principles of the design. This corresponds to Rasmussens's (1986) computer repair example, where the troubleshooting technician is shown to vary the resolution (aggregation) and his attention span (abstraction) to focus on particular aspects of the system.

The nested control structure of TECS has been captured by introducing levels of control sophistication that correspond to the identified control problems. Each control problem was mapped onto its own abstraction hierarchy, and the abstraction hierarchies were stacked to represent the nested control structure of the whole system. The resulting abstraction–sophistication analysis has been shown in Figure 4.9.

The separation of the control problems for separate analysis does not mean that the corresponding system parts can be designed independent from each other. For example, all the gain and limiter values throughout all levels of control sophistication are interdependent and need to be selected to preserve the energy relationship in the signals and achieve the 'energy based control decoupling'. Not only does the magnitude of the signals in the two signal paths need to be matched but also their rate of change. These constraints need to be defined at the interfaces between the levels of control sophistication to ensure the design's consistency across the levels.

It has not been found possible to model the functioning of TECS with the abstraction hierarchy. With the accompanying text, the abstraction hierarchy captures the design of TECS with the focus on the reasons for design. Links are made between the control diagram and the formulated design goals.

The abstraction hierarchy is a framework to represent knowledge and to link knowledge with means–ends links. These links represent the reasons for design features from a designer's perspective. The abstraction hierarchy allows the TECS control diagram to be included but the *language* of the control diagram is not part of the abstraction hierarchy. The ASA extends the abstraction hierarchy to make explicit the nested structure of the control problems.

In the next chapter, the ASA will be applied from the beginning of the design process, guiding the design and development of a complex automated system. Further research into the representation of nested structure of control problems in WDA is presented.

5

Case study: Design of a mini–UAV system

In the previous chapters the longitudinal flight control task was analyzed, first for manual control and then for automated control. The analysis of Chapter 3 revealed the energy constraints imposed by the physics of flight. The analysis of Chapter 4 showed how the Total Energy Control System (TECS) was designed to deal with them. The analysis also revealed that more constraints than just the energy management principles were part of the *design territory* of TECS. Among those were control theoretical considerations and constraints imposed by ATC.

To further develop Work Domain Analysis (WDA) for Ecological Automation Design(EAD), its application during the design and development of a new, highly automated mini Unmanned Aerial Vehicle (UAV) system is studied. The analysis of flight and flight control is no longer sufficient to understand the complete design territory (or designers work domain) of a UAV system. For example, navigation by itself can be regarded as a work domain, while it is only part of the work domain of the UAV system and independent of flight. Additionally, mission representations cannot be derived from the physics of flight, although flight capabilities do impose constraints on mission capabilities.

This chapter focuses on the use of WDA to keep track of the constraints that are introduced by the designed automation. We demonstrate that WDA can be adapted to allows us to manage the introduced constraints with the purpose to limit the automation introduced complexity. A number of the many design considerations of our UAV system, SmartUAV, are described using the abstraction–sophistication analysis.

5.1 Introduction

In 2005 the D–CIS Lab and ASTI (Aerospace Software and Technologies Institute) started a collaborative project to design and build a mini–UAV system: SmartUAV. SmartUAV is a fully functional mini–UAV system that is capable of controlling multiple mini–UAVs. The system serves as a technology demonstrator and as a platform for embedded systems research for higher level control such as: autonomous decision making, swarming and multiple platform payload coordination.

Four aspects of the mini–UAV system make it relevant to Ecological Automation Design (EAD). First, as stated in Chapter 1, UAV technology is pushing the boundary of automation design. Second, as explained in Chapters 1 and 2, the design of a highly automated system faces challenges with respect to supporting human control of the system. Third, SmartUAV is designed from the ground up, giving the designers maximum freedom in defining the entire system. Fourth, it is a fully functional embedded system, and "embedded systems design requires a more holistic approach that integrates essential paradigms from hardware and software theory" (Henzinger & Sifakis, 2007), which calls for EAD. SmartUAV has the ingredients to become a very complex system and its development forms an interesting case of the further development of EAD.

This chapter shows the process of applying the Abstraction–Sophistication Analysis (ASA) framework to map out the design space for SmartUAV. Over ten people with various backgrounds and interest have contributed to the project over a period of more than four years. Most contributors were concerned primarily with the engineering achievements, and less with human factors and the ecological approach. Therefore, the ecological automation design principles being developed during the project could be applied only to parts of the system. Their application mainly impacted the larger picture, and had less impact on the detailed design choices. The descriptions in this chapter focus on those aspects of the design that have been affected.

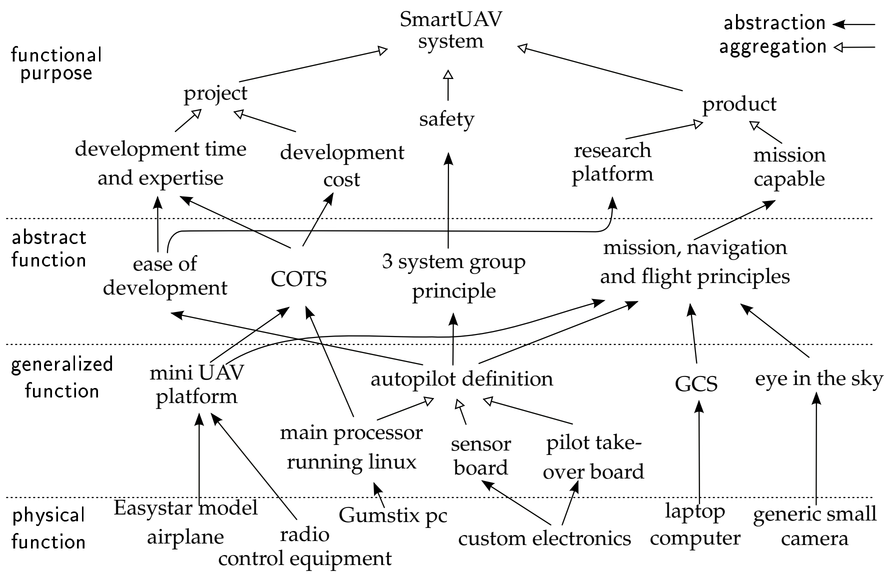


Figure 5.1: A first chunking of the UAV project, showing the functions that the system should provide as first requirement in the abstraction hierarchy. Solid–head arrows show means-ends relationships and hollow-head arrows show aggregation relationships.

5.2 System definition

The distinction is made between three types of constraints that shape the system design process. First are the constraints that the project imposes on the design, e.g., budget. Second are constraints that are part of the environment that belongs to the work domain but not to the system, as illustrated in Figure 2.1. Third are the constraints that are imposed by the system and the design choices. This section describes the first type of constraints, thus how the project and the initial system requirements constrain the design process. These constraints can also be structured using the abstraction hierarchy. Figure 5.1 shows a *first chunking* of the initial SmartUAV system and project tradeoffs. Hollow–head arrows represent aggregation at one level of abstraction, and solid–head arrows represent means–ends relationships across levels of abstraction. The initial, top level, requirements are discussed according to the decomposition at the functional purpose level: project, safety, and product.

5.2.1 Project constraints

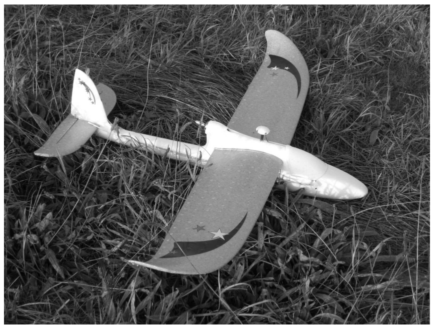
Project requirements shape the final system and as such they are part of the designer's work domain. The three main project constraints are the limitation of the cost, the needed development time and the required expertise. Therefore, commercial off the shelve (COTS) components are used where possible to reduce the expertise and the development time that is needed. The use of COTS components is regarded as an organizing principle of the project and can be found on the abstract function level. As Figure 5.1 shows, the COTS 'Easystar model airplane' used as the 'mini UAV platform', and the COTS 'Gumstix pc' used as an autopilot are the main contributors to achieve the cost and time saving COTS principle.

The 'ease of development' organizing principle is achieved by implementing the main processor of the autopilot as a 'Gumstix pc' that runs the mainstream linux operating system. Basic general programming skills can be expected from embedded systems designers but the need for hardware specific programming should be avoided for the end users of the research platform. Therefore, apart from one microprocessor located on the sensor board, the UAV autopilot can be programmed using the common C++ programming language. The microprocessor handles the sensor data and is not subject to much reprogramming.

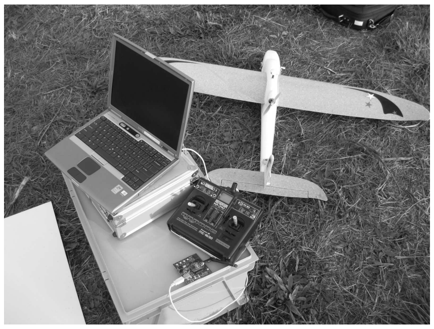
5.2.2 Purpose of SmartUAV

SmartUAV supports different types of UAVs. Two types of UAV platforms are exemplified in this chapter: the real–world Easystar fixed wing airplane, and a simulated UAV helicopter platform. Figure 5.2 shows the Multiplex Easystar. The mini–UAV system is designed to have mission capabilities based on the EMAV and IMAV outdoor flight competitions that are exemplified by the EMAV09 outdoor competition rules (EMAV09, 2009). A typical mission involves identifying targets, locating a vehicle in a search zone, fly through arches that represent an urban canyon, dropping a sensor in a drop zone, and a precision landing.

The competitions stimulate participants to generate innovative solutions to flight control, navigation, and especially mission execution. A such, SmartUAV is set up to allow adaptation to specific applications, supporting flexible flight, navigation and mission solutions. Therefore, 'mission, navigation and flight principles' are represented on the abstract function level. This term represents the various control problems that will be discussed later in



(a) The Easystar model airplane that is used as the flying test platform for the mini UAV system.



(b) The Easystar, a laptop computer as the GCS, the long range modem, and the safety pilot's radio transmitter.

Figure 5.2: Easystar airplane and ground equipment during flight tests in 2009.

this chapter. These are, in turn, achieved through the 'mini–UAV platform', 'autopilot definition', the 'GCS' and 'eye in the sky' concepts on the generalized function level.

The 'GCS' represents the on–ground hardware needed for control over the mini–UAVs. It contains a laptop computer that runs the SmartUAV GCS control software that is being developed in–house. It provides the user interface and part of the mission and navigation solutions. Three main configurations are supported by the software: real flight, hardware in the loop simulation and fully simulated. The latter mode allows rapid development of automation, control and interface concepts.

5.2.3 Safety requirement

Safety of the flying platform is a main design requirement. The system needs to be safe, and at the same time support experimental software and hardware. Therefore, a flight critical part of the hardware is defined that ensures robustness and stability. The design has been made with and reviewed by electronics experts from Thales Nederland BV. The flight–critical software is kept simple and unmodified after it is extensively tested.

Legislation requires that a safety pilot is standing by and can take over manual control at all times. In the event of a complete autopilot failure, the UAV can be flown like a recreational model airplane and be operated under model–aircraft regulations. This construction is mainly due to the fact

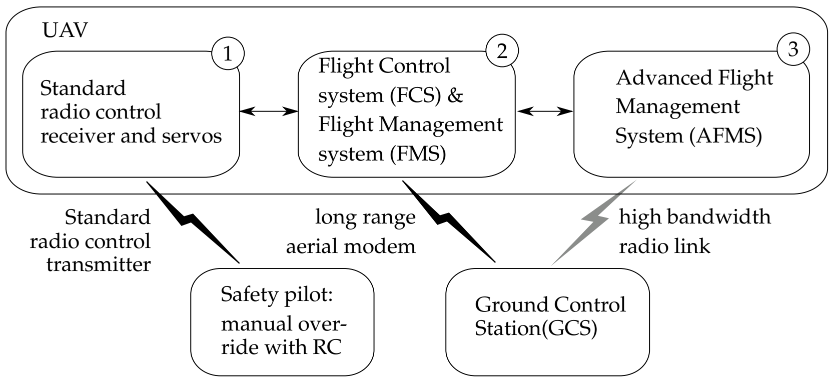

UAV
1
Standard radio control receiver and servos
2
Flight Control system (FCS) & Flight Management system (FMS)
3
Advanced Flight Management System (AFMS)
Standard radio control transmitter
long range aerial modem
high bandwidth radio link
Safety pilot: manual override with RC
Ground Control Station(GCS)

Figure 5.3: Three system group principle. The three system groups are implemented independently. When the experimental system group 3 fails, the more robust system group 2 can take over. System group 1 is based on simple and proven radio control technology, allowing a safety pilot to take over control at all times.

that regulations for employing UAVs were non–existent when the project started, and are currently still premature. The stringent safety requirements should not limit the 'product' purpose of the system: 'research platform'. Therefore, to organize the system, the 'three system group principle' was defined at the start of the project. It is represented at the abstract function level as a system organizing principle. The UAV control hardware and software are divided in three systems groups as shown in Figure 5.3.

System group 1 contains the standard radio control (RC) equipment that allows the UAV to be flown like a regular model aircraft by the safety pilot in case system group 2 fails. System group 2 contains the flight critical Flight Control System (FCS) and the Flight Management System (FMS) that provide flight control and basic navigation. The FMS's main task is to return to the home waypoint when system group 3 fails or when the data link with the GCS is lost. System group 3 contains the Advanced Flight Management System (AFMS) that is a separate computer board for experimental navigation and collision avoidance algorithms. The GCS is not put in one of the three system groups since it is on the ground, and as a research system, the operators have the ability to troubleshoot and solve problems. For example, a mid–flight reboot of the GCS is possible due to the integrity of system group 2.

5.2.4 Autopilot hardware definition

The hardware architecture was defined at the beginning of the project and was a direct result of the definition of the three system group principle. Figure 5.4 shows the basic hardware architecture of the UAV system and to which system groups the components belong.

The MAVPilot 3

The MAVPilot 3 consists of a custom designed sensor board and a COTS Linux based computer that runs the FCS and FMS in separate threads. The sensor board has the sensors needed for flight control with the option to connect additional sensors. These are: the 6 degrees of freedom Inertial Measurement Unit (IMU), a GPS receiver, magnetometers, a barometric altimeter and a differential pressure sensor to connect a pitot tube for airspeed measurement. The sensor data is preprocessed by a dedicated IO processor to unload the main processor from this task. A long range aerial modem is used as a two–way data link between the UAV and GCS.

The *servo control center* can switch control over all servos between the FCS and the safety pilot to facilitate manual take over in emergencies or during testing. In principle the safety pilot controls the manual take over switch through one RC channel.

The FCS and FMS threads run on the main processor. The FCS reads sensor data from the IO processor, controls communication with the GCS over the long range modem and controls the aircraft's flight control actuators. Additionally, the FCS controls the UAV to fly the commanded velocity vector. The FCS receives the commanded velocity vector from either the FMS or AFMS, as visualized with the switch in Figure 5.4. The FMS monitors proper AFMS functioning and controls the switch to take over flight control in that case of an AMFS failure. The FMS provides basic navigation functionality to be able to return to the home waypoint.

The advanced flight management computer

The AFMC is the Advanced Flight Management Computer, a separate computer board, on which the AFMS runs. It runs advanced navigation functions, collision avoidance algorithms, mission logic, and can process camera images for on–board vision. The AFMS is used to implement any functionality that is more advanced than the basic FMS functionality. These are

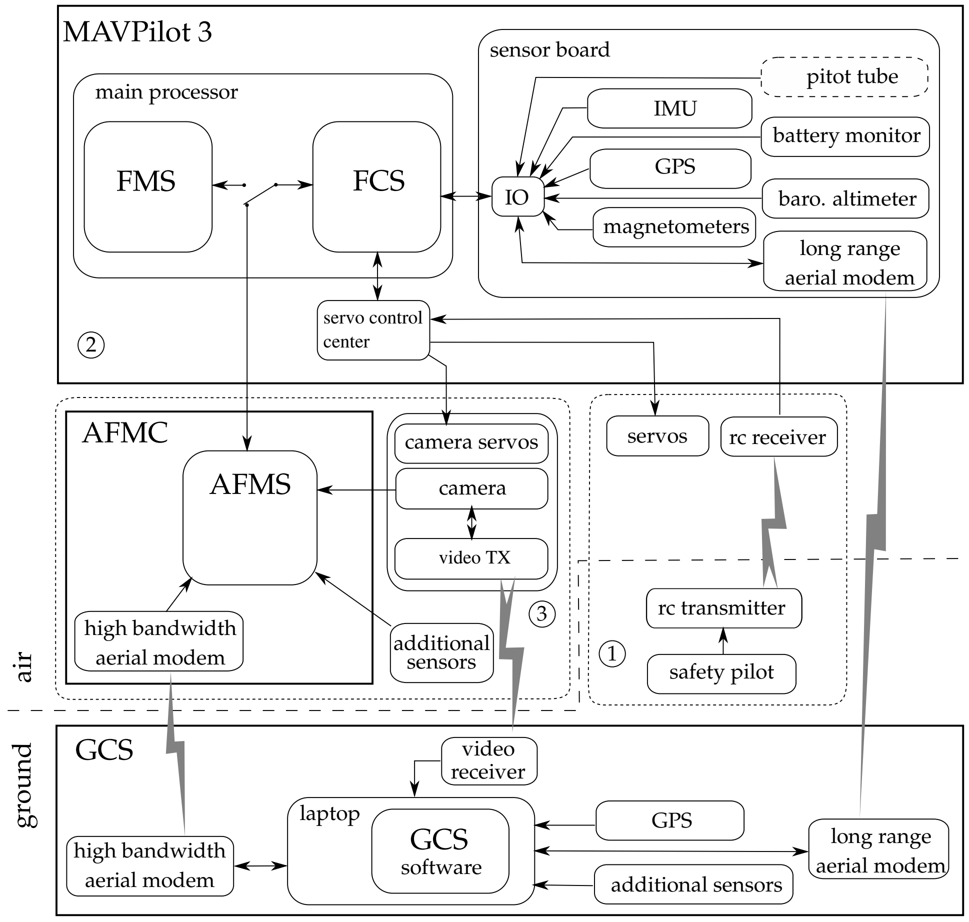


Figure 5.4: The UAV hardware architecture, showing the three system groups and to which group the main components of the UAV control system belong.

of a more experimental nature and therefore the AFMC belongs to system group 3. The design foresees a short range but high bandwidth link that will be used for bulk data communication (e.g., images or video) between the GCS and AFMS. The AFMS bypasses the FMS and sends velocity vector commands directly to the FCS.

5.3 The levels of control sophistication

The first step of WDA for EAD, is the delimitation of the work domain (Rasmussen et al., 1994; Vicente, 1999). Our work domain includes the complete UAV system. The scope of the analysis is significantly larger than the analysis presented by Castro and Pritchett (2005) for the same domain but without the nested abstraction hierarchies. The delimitation of the work domain can be visualized using the *first chunking* that was discussed above. The focus of the analysis is highlighted with a contour in Figure 5.5, showing that the analysis will focus on the 'product' purpose. The contour represents the designer's *work domain* for the following discussions in this chapter. We emphasize the designer's perspective of the analysis, and that WDA is used to generate a map of the constraints on the design.

After delimitation of the work domain, the nested structure of the control problems is identified and used to define the levels of control sophistication. SmartUAV is divided in six nested control problems: *flight*, *flight control*, *aviate*, *navigate*, *mission*, and *joint mission*. Below the general nature of each level is discussed in the light of automated control but independent of its implementation. In the next section they will be discussed in detail for the SmartUAV system.

Flight is a prerequisite for UAV operations. Flight is achieved by a flying platform that can have a number of different forms. For mini–UAVs, the most common ones are: fixed wing, rotary wing in classical helicopter configuration, and rotary wing in quad rotor configuration. Flight is mainly achieved through mechanical and aerodynamic design. To the designer, the capabilities of the platform (e.g., hover capability) and its performance (e.g., endurance) are the main constraints.

Next is **flight control**. Once flight has been achieved, it needs to be controlled. This includes electronic stabilization of the platform (if it is not inherently stable) and the ability to fly in the desired direction. This is typically achieved through electronic measurement and computer processing.

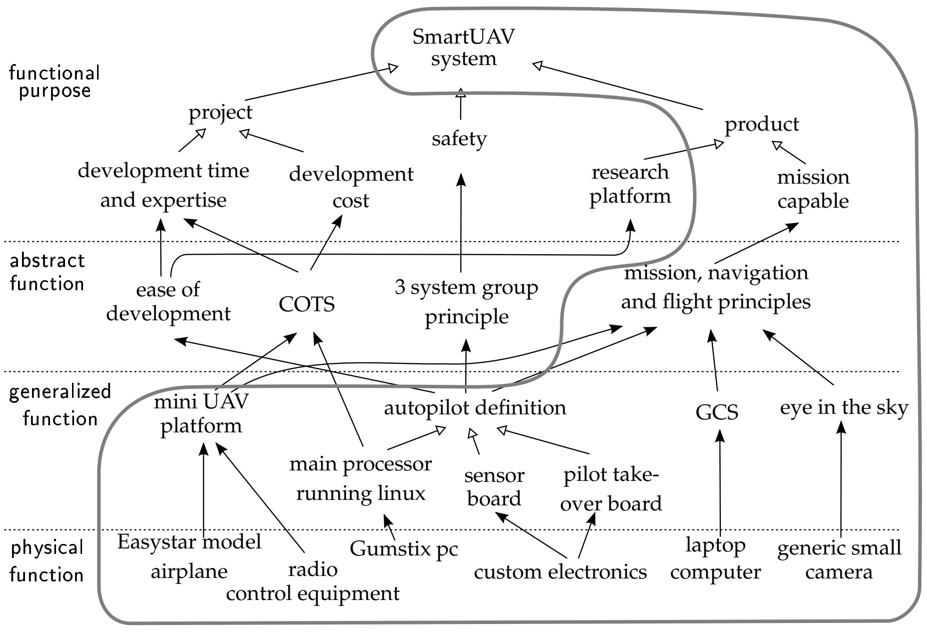


Figure 5.5: The first chunking of the UAV project with the solid contour showing the area of the work domain that is focussed on.

The flight control will alter the initial capabilities and performance by staying well within the boundary of the flight envelope, e.g., by limiting manoeuvring rates. To the designer, the main constraints are the dynamics and performance envelope of the combination of the flight control solution and the platform.

With the ability to fly in the desired direction, the UAV needs to 'know' where to go, and where not to go. In piloting terms this is to **aviate**. This control problem requires a representation of what is a safe and desired field for travel (Gibson & Crooks, 1938). It includes *collision avoidance*, which is a main challenge in current UAV automation.

When this control problem is left unsolved by automation, the system operators are required to control this. A crude and commonly used solution is to ensure that the UAV operates in airspace that is closed for other air traffic, and at an altitude that clears all ground obstacles. By identifying this control problem prior to automating the UAV system, a place for future automated solutions is reserved and potential patchwork after automation design is avoided.

Navigation is not specific to UAV control and the concepts well established in the aviation domain are adopted. Flight plans and waypoints are the language of this control problem. They deal with a bigger picture than the *aviate* control problem. A flight plan can be made to avoid flight over or past buildings but does not take enroute obstacle avoidance into account and other possible deviations from the initial flight plan.

A navigating UAV can fly to points of interest and the **mission** addresses the useful functions it can perform at the points of interest. Control of missions is a high level control problem with its own representations for mission objectives and possibly mission success. Mission control involves the coordination of the UAV as a flying platform and its payload. In our examples, a pan–tilt camera is the main payload. Its orientation with respect to the airframe needs to be coordinated with flight direction and manoeuvering.

Finally, the **joint mission** level is identified as a placeholder for representations of collaborating UAVs. Although not further detailed, at this level representations for multi–UAV entities (e.g., swarms) and collaborative efforts can be defined.

The above control problems form the definition of the *levels of control sophistication*. They form a hierarchy of nested work domains, shown in Figure 5.6. Higher levels depend on the functioning of lower levels. Each level of control sophistication describes a specific control problem and has its own boundary. The inner–most levels have the smallest delimitation and the outer–most levels have the largest delimitations. The inner levels fall within the delimitation of the outer levels, and the outer levels include the designed automation at the inner levels as part of their work domain. Moving outwards, the levels deal with higher order control problems, each with its specific representation. The levels are chosen such that each level represents (automated) control functions that the higher levels rely on. At each level, automated functions are introduced and they become part of the work domain of the adjacent higher level.

The levels of control sophistication are chosen according to the control problems that are identified in the operator's work domain, they are not based on hardware choices. Figure 5.7 shows how the levels of control sophistication map onto the hardware architecture. The considerable overlap shows that automated processes belonging to different levels of control sophistication are implemented on the same hardware. Figure 5.7 also shows that the

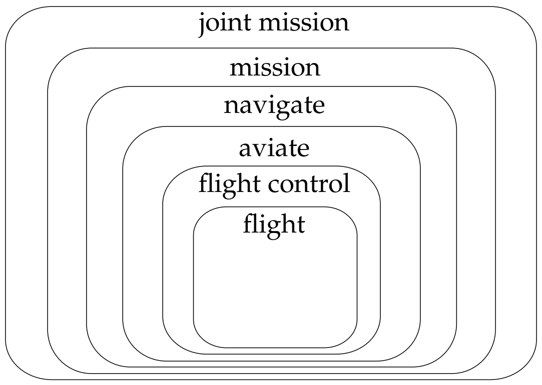


Figure 5.6: The levels of control sophistication represent the nested structure of the control problems in the designer's work domain.

FMS and FCS both belong to a single level of control sophistication because they belong to flight critical system group 1, which is isolated for safety, in accordance with the 3 system groups principle).

5.4 Abstraction–sophistication analysis

In the abstraction–sophistication analysis, the abstraction hierarchy is applied at each level of control sophistication. This maps out the constraints that need to be taken into account per control problem. Rasmussen's (1986) top four levels of abstraction are adopted. To avoid descriptions in great detail, the physical form level is omitted and only the top four levels of abstraction are used.
The bottom level, the physical form level, is largely shared between all levels of control sophistication. Each level of control sophistication gives a specific direction to the abstractions that are made in the abstraction hierarchy. For example, looking at the generalized function level of abstraction, a building presents an obstacle to avoid at the aviate level while it would represent an observable target at the mission level.

In the following, the design considerations and control solutions are discussed per level of control sophistication and per level of abstraction for the design space of the UAV system. A considerable amount of detail is involved in the discussions, which cannot be avoided since it is the purpose

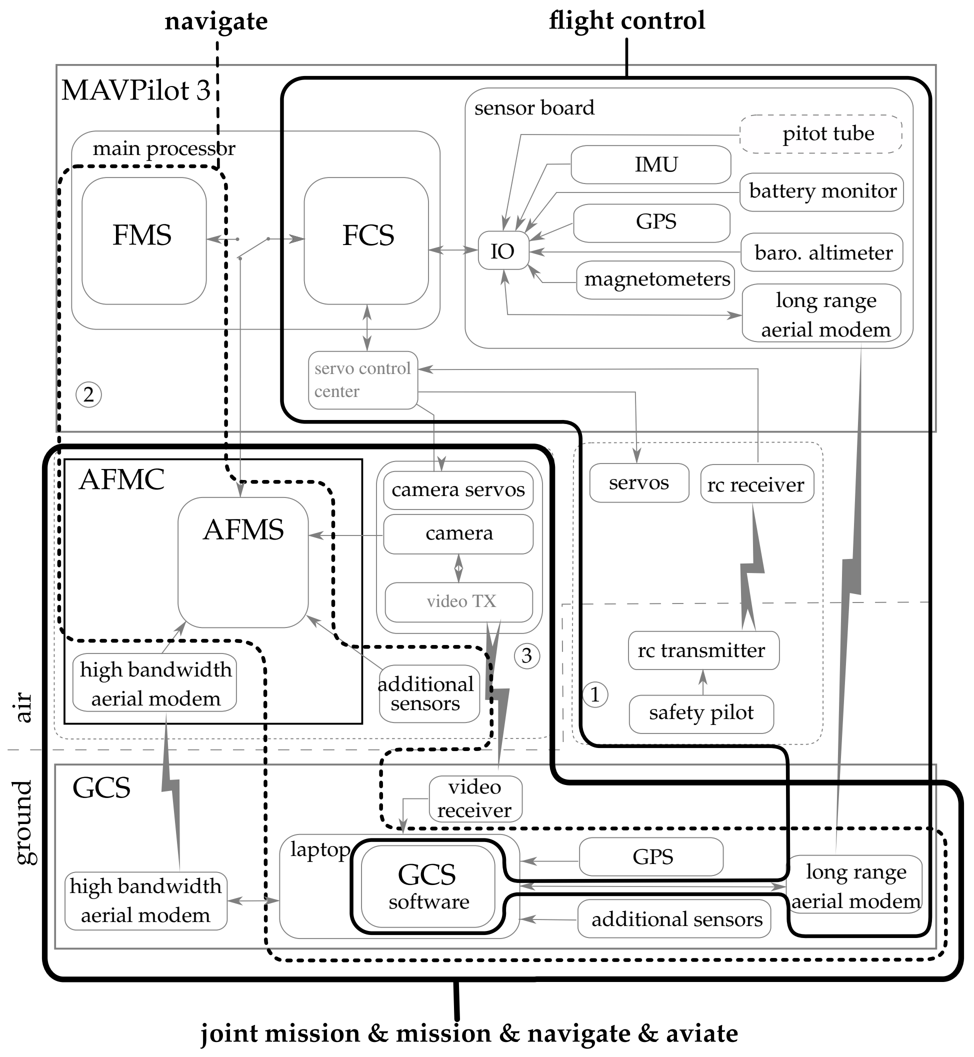


Figure 5.7: The UAV hardware architecture and contours that show to which levels of control sophistication the components of the architecture belong. The chaotic nature of the diagram and the overlap of the contours show that there is not a clear mapping between the components of the architecture and the levels of control sophistication. This is to be expected since the levels of control sophistication represent a functional decomposition rather than a structural decomposition.

of WDA to link the many details in a work domain to less but more abstract principles and purposes. Without the details, this cannot be demonstrated. The levels of control sophistication are discussed top–down because it will give the reader a better introduction to why the technology on the lower levels is implemented as it is. At each level of control sophistication, the levels of abstraction are also discussed in top–down fashion. This choice has been made because it will allow us to first explain the more meaningful concepts and functions in the work domain and then zoom in on the more concrete details that implement the higher level functions.

5.4.1 The joint mission level

The joint mission level is the first and outer–most level of this analysis. This level is a placeholder for joint control of multiple UAVs and/or other entities. The principles have not been developed or implemented as part of this work but could include concepts that allow a group of mini–UAVs to be controlled as a single entity with some exceptional properties that are only available through joint coordination of UAVs. Without going into the details of how joint UAV coordination should be implemented, it is assumed that each individual UAV should be capable of performing a mission. The following levels of control sophistication are described in more detail.

5.4.2 The mission level

The mission level deals with single–vehicle mission control and has representations of mission objectives. In the current system, a mission can be defined using mission elements. The mission objectives supported by Smart-UAV are directly based on the I/EMAV flight competitions, which are described in (EMAV09, 2009). Mission objectives common to the competitions are: take–off, identify object, drop paintball, locate object, fly through an urban canyon, search for an object, and precision landing.

Before a flight, the defined mission is converted to a flight plan that can be executed by the navigation logic. The drawback of this approach is that mission elements are not implemented as active elements and cannot control payload in real time. To circumvent this problem, actions for payload control were assigned to waypoints that signify a position at which the payload needs to be active. However, this proves to be a contamination of the navigation level with mission specific functions, which is contrary to the ap-

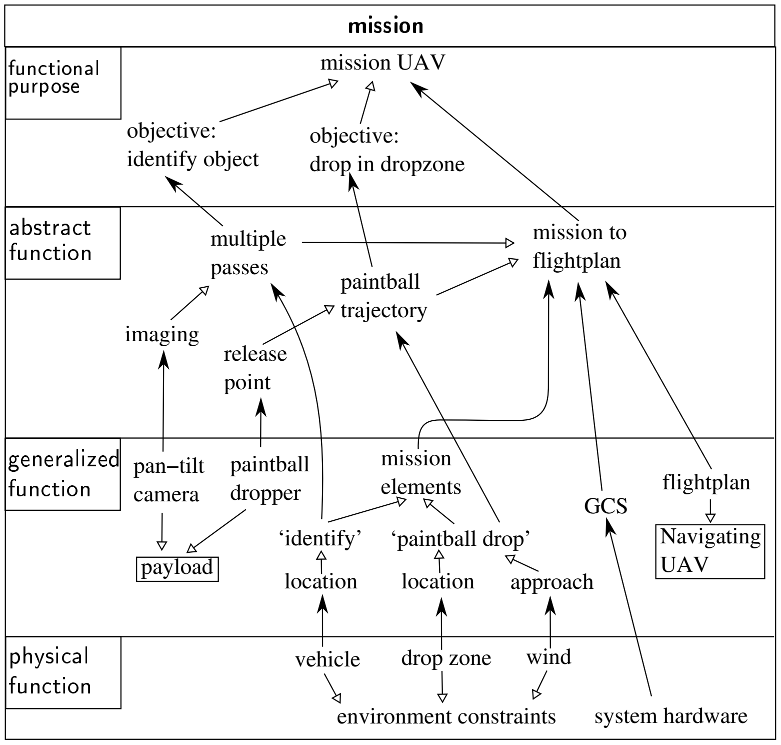


Figure 5.8: Graphical representation of the abstraction hierarchy at the mission level.

proach taken with the ASA. Therefore, real–time mission control is planned for future implementations of the AFMS. Despite the drawback, the current implementation is used to exemplify the analysis.

The payload is considered as a separate subsystem and is not included on lower levels of the analysis. As a separate system, the payload can be analyzed with its own specific levels of control sophistication. This chapter, however, focuses on the analysis of the UAV system only. The functionality of a pan–tilt camera is used at the mission level but not further analyzed. Figure 5.8 shows the graphical representation of the abstraction hierarchy at the mission level of control sophistication. The levels are discussed below.

Functional purpose

Two mission objectives are exemplified: the identification of an object, and the drop of a paintball in a drop zone. The first mission element requires a clear image to be presented of a stationary object in a known location. Control over the UAV's flight and the pan–tilt camera need to be coordinated. The second mission element requires a paintball to be dropped close to a predetermined point. Again, the control over the UAV's flight and the paintball release mechanism need to be coordinated. This is typical of the mission level.

Abstract function

The abstract function level represents the transformations from generalized mission elements (at the generalized function level) to the functional purpose. In the current implementation, this means the conversion from a mission definition to the flight plan. Therefore, in Figure 5.8, the 'paintball release' and 'multiple passes' are shown to be part of 'mission to flight plan'. The flight plan holds the information for flight and payload coordination.

For the 'identify mission' element, a number of passes over the location of the object is generated from different directions. As the UAV passes the object location, pictures are taken that can be used to identify the object. The pan–tilt camera system tracks the object location based on GPS.

For the paintball drop mission element a single pass is generated from the desired approach direction at the desired altitude. Taking wind, airspeed, altitude and the paintball drag into account, a release point is calculated. The paintball is released when the UAV passes through the release point.

These conversions are the organizing principles of the mission level. They are artificial in the sense that they are introduced by the system designers, yet they determine system behavior. They are the representations for reaching the stated objectives.

Generalized function

At the generalized function level, descriptions of the mission elements are given as the generalized building blocks for complete missions. A mission always starts with a takeoff and ends with a landing. In between, the other mission elements can be planned in any configuration. The operator plans a mission on the GCS map through the graphical interface supporting the mission elements directly. In this way the operator can directly see the UAV behavior in the context of the mission it is flying. It prevents the operator from having to view the mission in terms of waypoints and the flight plan, which contains much more detail but lacks the meaning of what the UAV is doing.

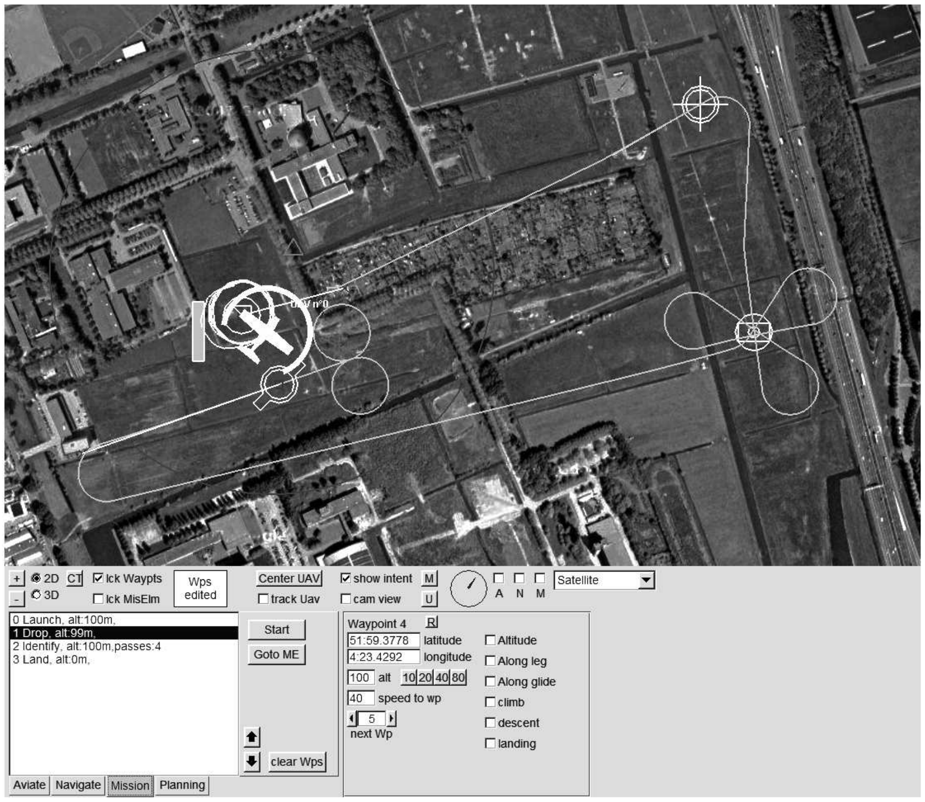
The GCS is represented at this level, and as shown with the abstraction arrow, it achieves the 'mission to flight plan conversion'. A graphical representation of a mission at the GCS interface is shown in Figure 5.9. The 'identify' mission element is defined by the location, the number of passes it will fly over the object location, how long each pass should be, and the altitude. The paintball drop mission element is defined by the location, the altitude of the drop, approach heading, and approach length. These parameters can all be adjusted using the user interface of the GCS.

Physical function

In the above descriptions, the concepts were abstract and generalized. At this level the physical function of the elements are represented. A high level of aggregation has been chosen to prevent countless details from cluttering this level. The mission is, in the end, designed to do something meaningful in the physical world. In this example, the to–be–identified object is a (stationary) vehicle. Its location is meaningful to the generalized function level but its exact characteristics are not taken into account. Similarly the drop zone's location is meaningful, and the wind direction is taken into account to set the approach heading.

5.4.3 The navigate level

The third level is the navigate level. At this level the navigation control problem is represented. Navigation is the art of measuring position, course, speed, distance traveled, with the goal to travel from one place to another. SmartUAV, like many modern systems, uses GPS to measure the UAV's position, speed, and course. It uses waypoints as means to define a route or flight plan. These are familiar concepts that will be mapped onto the four levels of the abstraction hierarchy. As will become clear, the navigate level relies on the aviate level for sense and avoid. Figure 5.10 shows the graphical representation of the abstraction hierarchy.



[✓] lck Waypts
[ ] lck MisElm
[✓] show intent
[ ] cam view
Satellite

0 Launch, alt:100m,
1 Drop, alt:99m,
2 Identify, alt:100m,passes:4
3 Land, alt:0m,

Waypoint 4
51:59.3778 latitude
4:23.4292 longitude
100 alt
40 speed to wp
5 next Wp

[ ] Altitude
[ ] Along leg
[ ] Along glide
[ ] climb
[ ] descent
[ ] landing

Figure 5.9: A mission is shown on the map of the UAV GCS. It consists of the following mission elements: launch, paintball drop, identify, and land. The UAV is in a circular holding pattern above the launch area. The predicted flight path is shown for the mission.

Functional purpose

The functional purpose is to provide navigation functions in terms of flight plan execution. The purpose is to have a navigating UAV but how is that defined? The lower levels of abstraction detail this purpose.

Abstract function

As shown in Figure 5.10, the navigate level has two implementations. One is based on the FMS, and the other is based on the AFMS. The current operation of the Easystar uses the FMS for it 'basic fligh plan execution logic'. The

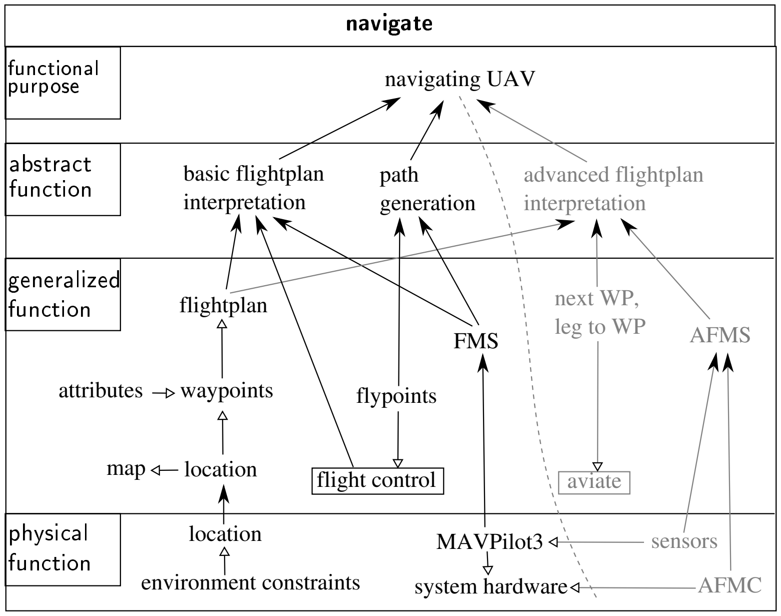


Figure 5.10: Graphical representation of the abstraction hierarchy at the navigate level. The dashed line separates the two different implementations. Left of the dashed line shows the navigation based on the FMS without any implementation of the aviate level (current Easystar system). Right of the dashed line shows the navigation based on the AFMS with an implementation of the aviate level (simulated helicopter).

use of the AFMS for 'advanced flight plan execution' is planned for future implementations.

The 'basic flight plan execution' will generate flight path commands that are directly sent to the flight control logic, bypassing the aviate level. The flight path commands are based on the waypoint in the flight plan. Each waypoint is defined by a position and altitude, and a number of attributes that further set the behavior of the navigation logic:

- **Speed**: sets speed to this waypoint.
- **Next waypoint**: sets the waypoint to fly to after this waypoint.
- **Altitude**: instructs the UAV to reach this waypoint's set altitude by circling at the waypoint, before continuing to the next one.
- **Along leg**: instructs the UAV to fly along the lateral leg connecting the

previous and this waypoint.

- **Along glide**: instructs the UAV to control the vertical flight path along the leg that connects the previous and current waypoints.
- **Climb**: instructs the UAV to climb with maximum power if this waypoint is higher.
- **Descent**: instructs the UAV to descent with minimum power if this waypoint is lower.
- **Landing**: instructs the UAV that this waypoint marks the landing spot, thus to cut throttle and to flare near the ground altitude.

The 'advanced flight plan execution' works in conjunction with aviate level logic. Instead of generating flight path commands , the navigation logic only determines what the next waypoint is, and passes it on the aviate logic. The aviate logic will then calculate a collision–free path to that waypoint.

Generalized function

The concepts at the generalized function level can best be described by their representation at the interface of the GCS. These are the concepts for UAV navigation that the operator is most familiar with. They are the flight plan, the waypoints it consists of, and the map of the environment. The operator can define a flight plan by creating waypoints, or can modify the flight plan that was generated by the mission logic from a mission definition. This allows the operator to verify and optimize the flight plan, and possibly take constraints into account that were unknown to the automated system (e.g., temporary no fly zones, or preferred routes).

To help the operator understand the behavior of the UAV, the expected flight path is calculated by a ground based simulation and represented as a predicted trajectory. It allows the operator to quickly see if the UAV's expected flight path matches his intentions. The generated path is commanded to the FCS in by a stream of *flypoints*. The flypoints were introduced to specify points that were less significant than waypoints in terms of navigation and could be generated in a stream to define a flight path. This proved especially useful for path planning at the aviate level that is discussed below.

All flight plan generation and modification activities take place on the GCS. After the flight plan has been made, either manually or automatically (by the mission level), it is uploaded to the FMS. After the FMS has properly received the flight plan it will proceed to execute it. The AFMS and FMS are represented at this level of abstraction and are shown to achieve the basic and advanced flight plan execution functions at the abstract function level in Figure 5.10.

Physical function

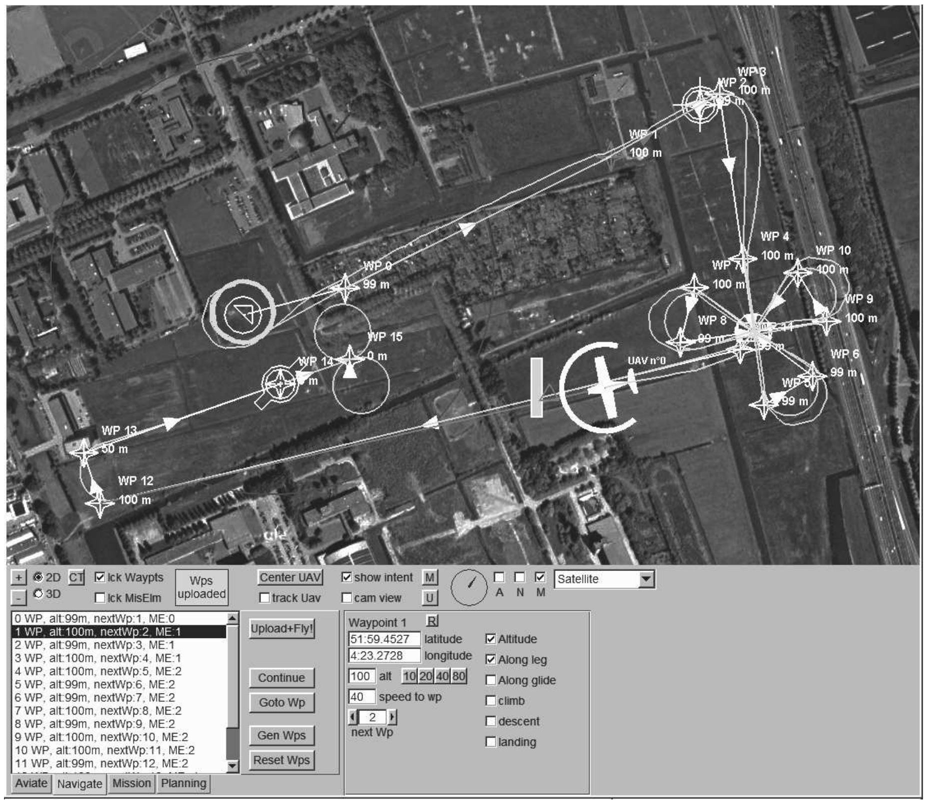
The physical function of a waypoint is the location on earth as opposed to a location on a map on the generalized function level. The system (computer) hardware that allows the FMS and AFMS to run is also defined at the physical function level. Other than that, the navigation level of control sophistication has little representation at the physical function level.

5.4.4 The aviate level

The fourth level is the aviate level. The control problem of this level is *knowing where to fly and where not to fly*. The flight plan provides the bigger picture and aviating or 'piloting common sense' focuses on local constraints. Deviating from the flight plan for collision avoidance is an example of control at this level.
Automation on the aviate level was not implemented in the Easystar platform. It was implemented in a simulated helicopter UAV only as a study and proof of concept. This implementation is used to exemplify the analysis at this level. It achieves autonomous avoidance of known obstacles stored in a database during simulated flight. It allows future addition of sensed or otherwise known static obstacles. Dynamic obstacles are not taken into account. Figure 5.12 shows the graphical representation of the abstraction hierarchy. The levels are discussed below.

Functional purpose

The purpose of the implementation is to give the helicopter UAV basic obstacle avoidance capability based on a database of obstacles and a path planning algorithm. The logic at the aviate level provides a safe and efficient path to the requested point (given that it is approachable), taking into account known obstacles and other information about the environment such as no–fly zones or preferred routes.



[✓ 2D] 3D
[✓] lck Waypts
[ ] lck MisElm
[ ] track Uav
[✓] show intent
[ ] cam view
Satellite

```
0 WP, alt:99m, nextWp:1, ME:0
1 WP, alt:100m, nextWp:2, ME:1
2 WP, alt:99m, nextWp:3, ME:1
3 WP, alt:100m, nextWp:4, ME:1
4 WP, alt:100m, nextWp:5, ME:2
5 WP, alt:99m, nextWp:6, ME:2
6 WP, alt:99m, nextWp:7, ME:2
7 WP, alt:100m, nextWp:8, ME:2
8 WP, alt:99m, nextWp:9, ME:2
9 WP, alt:100m, nextWp:10, ME:2
10 WP, alt:100m, nextWp:11, ME:2
11 WP, alt:99m, nextWp:12, ME:2
```

Waypoint 1 R
51:59.4527 latitude
4:23.2728 longitude
100 alt 10 20 40 80
40 speed to wp
2 next Wp

[✓] Altitude
[✓] Along leg
[ ] Along glide
[ ] climb
[ ] descent
[ ] landing

Figure 5.11: The same mission is shown as in Figure 5.9 but it is represented in terms of the waypoints that were automatically generated based on the mission elements. Both the mission elements and the flight plan's waypoints are visible. The UAV is shown flying the mission. The actual flight path and the predicted flight path are both shown.

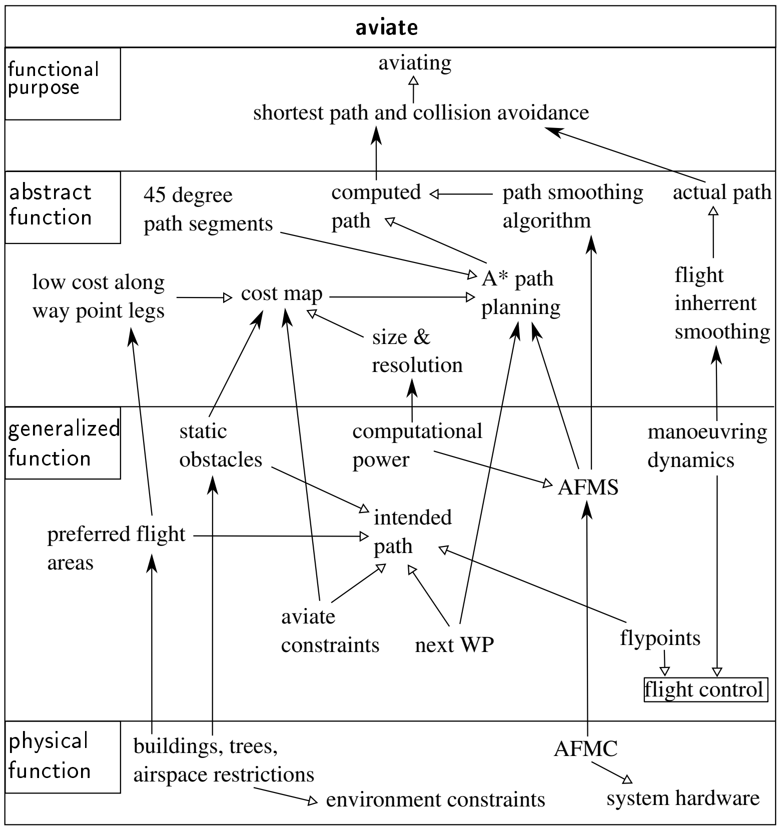


Figure 5.12: Graphical representation of the abstraction hierarchy at the aviate level.

Abstract function

The A* path planning algorithm is used for planning efficient paths around known obstacles (Oomkens, Amelink, Mulder, & van Paassen, 2008). A* works with a cost map and an algorithm that finds the cheapest path from one point to another based on this map. The cost map is a two–dimensional grid where each cell has a cost as an abstract representation of obstacles in the environment. The cost is high for cells that are undesirable locations for the UAV to be and low for neutral locations. The algorithm searches for a path on the cost map that has the lowest accumulated cost of all cells it passes through. From one cell the algorithm can move in eight directions; up, down, left, right, and the diagonals. Figure 5.13 gives an illustration of

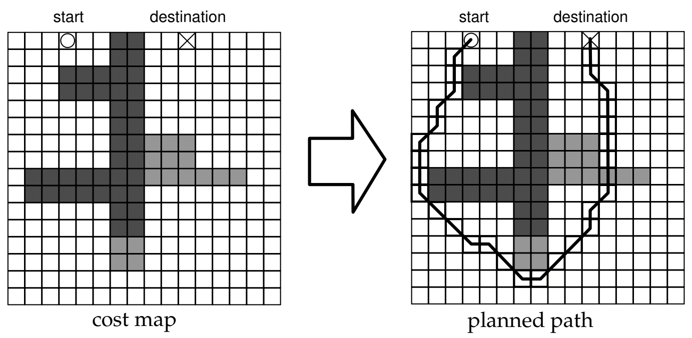


Figure 5.13: An example that illustrates A* path planning based on a cost map. The cost map is a grid that allows movement in 8 directions: horizontally, vertically and diagonally. The algorithm will find a path that is the cheapest in terms of the cost of the map tiles that the path occupies.

a cost map and a possible path found by the A* algorithm.

The grid introduces constraints on behavior at this level that will show up in the flown path. First, the world is viewed at a certain resolution to limit the computational time needed to calculate a solution. This impacts the fineness of the solution with the effect that some obstacles can clutter together and a possible path between those obstacles is not found. Second, the calculated path is made up from short line segments connecting adjacent cells that are at a multiple of 45° with each other. As a result the calculated path has a 'staircase' pattern that is purely an A* artifact and does not have any relation to the actual world, see Figure 5.13. Smoothing of the path is required to prevent the helicopter to from flying a visible staircase. The smoothing algorithm is also defined on this level. To prevent the smoothing algorithm from cutting corners through obstacles, obstacles were represented slightly larger on the cost map than in reality.

The A* algorithm is optimized to limit the number of possible paths that need to be explored to find the best one but computational load is still a big issue. Computational power is represented on the generalized function level, the constraints on computational power propagate up to the abstract function level. To limit the computational load the resolution of the raster can be reduced, the total search area can be kept small, and the number of solutions needed per unit of time can be limited. This has some implications for large maps but those have not been explored.

A full three–dimensional version of A* would increase the computational load to unpractical levels. Therefore, the altitude dimension has been approximated by two altitude levels. The altitude that splits the level is chosen to include the majority of obstacles (buildings, vegetation, etc.) in the lower level. The upper level includes only a few obstacles like the top of church towers. The level altitude was set at 50 meters in our simulation.

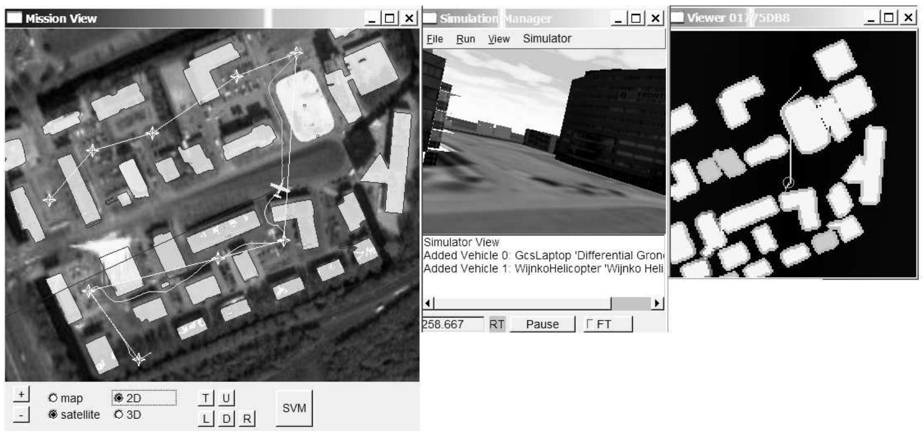
There is a distinction between the computed path and the actual path. They are both represented and the assumption is made that the actual path will be close enough to the computed path to be safe. The controlled UAV, as defined on the *flight control* level has to be able to follow the planned path accurately enough to avoid the obstacles along which the path is planned. For example if the flight controller has a low performance and large deviations from the planned path, collision avoidance may not work under all circumstances.

Additionally, the needed manoeuvring space of the UAV needs to be taken into account. In this case, the simulated helicopter would adjust its speed to be able to make sharp turns. Its ability to fly slowly and to hover allows the path planner to ignore the need for manoeuvering space, which would not be appropriate for a fixed wing platform that has a minimum turning radius. This illustrates how platform constraints on the flight level and the flight control level propagate up to this level.

Generalized function

At the generalized function level, the constraints that the operator can work with are represented. These are the constraints that define the A* cost map. Different types of constraints can be combined into the same cost map. Obstacles represent an infinite cost, while softer constraints, such as airspace above a city will represent a somewhat higher cost to avoid this area when possible.

The operator is also able to add constraints on the map. These are referred to as ‘aviate constraints’. The aviate constraints are not treated in any special way, they simply become part of the cost map. The advantage is that the path planning behavior will be consistent regardless of the type of constraint and the operator does not need to be aware of any special logic for ‘aviate constraints’. This allows different types of information to be taken into account without increasing the complexity of the automation logic. The operator does not need to understand the A* algorithms internal logic. Sim-



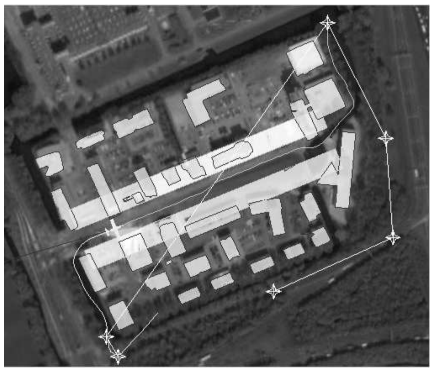
Simulator View
Added Vehicle 0: GcsLaptop 'Differential Gron
Added Vehicle 1: WijnkoHelicopter 'Wijnko Heli
258.667

Figure 5.14: The SmartUAV interface at the aviate level. The left window shows the map with the flight plan in terms of waypoints, the flown path, the computed path, and the obstacles as high cost areas on the map. The middle window is a simulated view from the UAV. The right window is a visual representation of the actual cost map that the algorithm uses.

ilar to the ant on the beach parable, discussed in Chapter 1, the planned path is easily understood based on seeing the constraints. The constraints that define the cost map are visualized in the interface and the operator can easily see, predict and understand the algorithm's solution based on the constraints.

An example of an operator definable shape is shown in Figure 5.15. The aviate constraint can be used to approach a point from a certain direction, for example, when approaching a building to look into a window. Other uses can be to block an airspace for the UAV to fly through, or to give preference to a certain path by locally reducing cost. For example, in the case of a forest fire, the airspace directly above and downwind of the fire can be blocked with an 'aviate constraint' to prevent the UAV from planning a path through hot and hazardous air.

The AFMS is represented here as the computation unit that runs the A* path planning logic. Its computational power will determine the resolution and size of the cost map at the abstract function level, as shown in Figure 5.10.


(a) Large rectangular shaped aviate constraints are place on the map by the operator to shape the future flight path of the UAV down the middle of this business park.

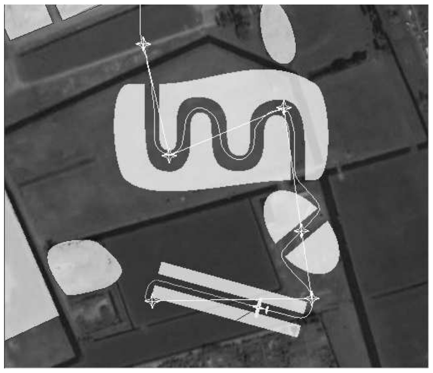
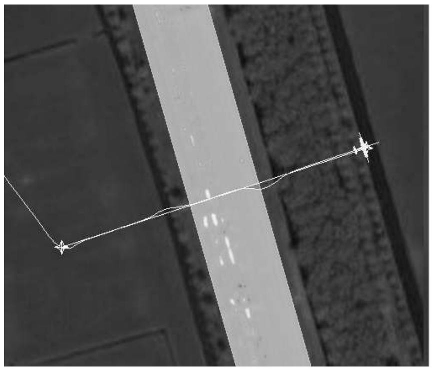

(b) Alternative shapes of aviate constraints are shown to illustrate the principle. They do not present obstacles but are a means for the operator to shape the UAV's flightpath. The algorithm's behavior is understood based on the aviate constraints and the waypoints.

Figure 5.15: Two examples of the use of aviate constraints.


(a) The two–dimensional aviate constraint has been placed over a highway.

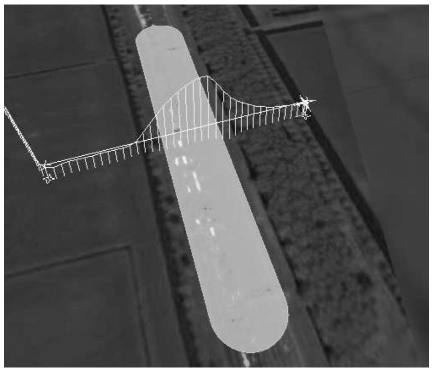

(b) This 3D view shows that the UAV has planned and flown a path over the aviate constraint, deviating to the higher flight level.

Figure 5.16: These simulations show how an aviate constraint can be used to prevent a low altitude flight over a highway. A path around the constraint is very long and the algorithms computes the lowest–cost path at the higher altitude over the aviate constraint. The UAV has a 'preference' to fly at the low altitude level because flight at high altitude level is constrained by a higher cost in the cost map.

Physical function

The physical functions of the environment that the UAV flies in, is expressed in terms of the objects that define obstacles at the generalized function level. These are physical objects such as buildings and trees, but also airports or other structures that represent restricted airspace. The system hardware at the aviate level is the AFMC that runs the AFMS thread. Although not further detailed, an in–depth analysis at this level would focus on power usage, heat production, and other physical properties of this computer board.

5.4.5 The flight control level

The fifth level is the flight control level that represents the flight control problem. Flight control is achieved by the autopilot, or by the human safety pilot in the case of an emergency. We continue describing the automatic flight control functions for the fixed wing Easystar platform.

In this analysis, the flight control and the flight control modes are represented in the same abstraction hierarchy, in contrast to the analysis of TECS in Chapter 4, where they were put on different levels of control sophistication. For UAV control, the focus is on the velocity vector control that facilitates the control on higher levels of control sophistication. The classical autopilot modes are added to give the operator direct control over the flight control logic, thus a means to interact at this level of control sophistication. Figure 5.17 shows the graphical representation of the abstraction hierarchy at this level of control sophistication. The levels are discussed below.

Functional purpose

The functional purpose is to achieve controlled flight. As shown in Figure 5.17, 'controlled flight' is decomposed in 'safety', 'product', and 'efficiency'. Safety has already been addressed in Section 5.2. Product represents the properties of the actual path flow, including its performance parameters. Efficiency is further decomposed into energy loss and wear. Excessive control activity will cause the servos and control surface hinges to be subject to excessive wear. Additionally, the available on–board electrical power will be used quicker due to excessive servo movements. Of the three functional purpose components, efficiency received the most attention to extend endurance and servo life.

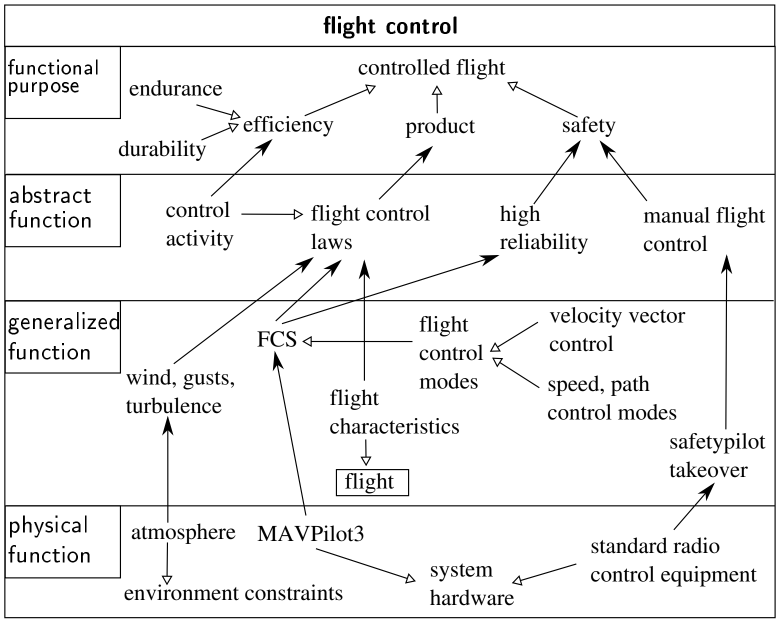


Figure 5.17: Graphical representation of the abstraction hierarchy at the flight control level.

Abstract function

Controlled flight can be achieved through either automatic flight control under normal circumstances or manual flight when the safety pilot takes over. For automatically controlled flight, the control laws define which variables are used for elevator, throttle, rudder and aileron control. Experimental trials led to the implementation of a control law that is a mix of the two previously described control strategies, the *throttle to speed and elevator to path* and the *elevator to speed and throttle to path* strategies, see Chapter 3 for their definitions. The resulting mixed strategy resulted in the most acceptable flight performance that takes the altitude–speed coupling into account, similar to what the energy control of TECS achieves (see Chapter 4) but without an explicit energy representation.

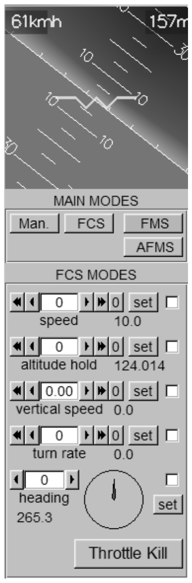


Figure 5.18: The UAV's instrument panel of the GCS interface is used to select and set the AFMS and FMS flight contol modes. In FMS mode, this panel provides basic flight controls modes: speed, altitude hold, vertical speed, turn rate, and heading hold. A throttle kill button allows (emergency) motor switch off.

Generalized function

The generalized function level deals with the concepts that the operator is the most familiar with. Operator's interactions with the flight control level, thus with the FCS, are mediated by the flight control modes. These are: speed, altitude hold, vertical speed, turn rate, and heading hold. The flight control modes provide the operator a relatively low level of flight control that bypasses the higher levels of control sophistication.

Physical function

The physical function of the autopilot is in terms of its power usage, heat production etc. The Radio Control (RC) equipment to allow a safety pilot to take over control, is also represented as system hardware. Not represented in Figure 5.17, but typically belonging to this level are: the electrical functions of the sensors, their signal to noise ratio, temperature bias, and other physical constraints that the sensors impose. RC equipment functions such as radio link range and noise sensitivity and servo characteristics like speed and travel are also typically represented at this level represented. However, in this design study the latter constraints are not studied to a great extent, they are simply properties of what is currently commercially available.

5.4.6 The flight level

The flight level is the sixth and inner–most level of control sophistication. It deals with the flying platform, the basis for the flying UAV. The choice of the flying platform determines what kind of missions can be flown. Its properties propagate up through the levels of control sophistication and ultimately affect the (joint) mission capabilities.

Because the Easystar is a COTS platform, the depth of the analysis on this level is limited. An abstraction hierarchy is given to show how the abstract requirements match the physical properties of the Easystar airplane. It is used to map how the choice of platform enables and/or limits functions on higher levels of control sophistication.

Note that the analysis of this level of control sophistication focuses on other constraints than the energy coupling that was discussed in the previous two chapters. The discussed properties were more relevant to the development of the whole system at the time. Figure 5.19 shows the graphical representation of the abstractions hierarchy. The levels are discussed below.

Functional purpose

The functional purpose of the flying platform is a trade–off because the ideal platform does not exist. From the available COTS platforms the Easystar had the most favorable characteristics for this project in terms of 'long enough endurance', 'payload capability', 'favorable dynamics', 'robustness', and 'environmental impact'. The choice for the Multiplex Easystar model airplane is largely based on the COTS requirement of the project, as shown in the first chunking in Figure 5.1.

The Easystar will fly up to 20 minutes carrying the autopilot and a camera payload. The Easystar is a model airplane for novice pilots implying that it is very stable and easy to control. It allows safety–pilot control take–over during development and simplifies flight control development. The Easystar is designed to absorb impact on less than perfect landings, which is ben-

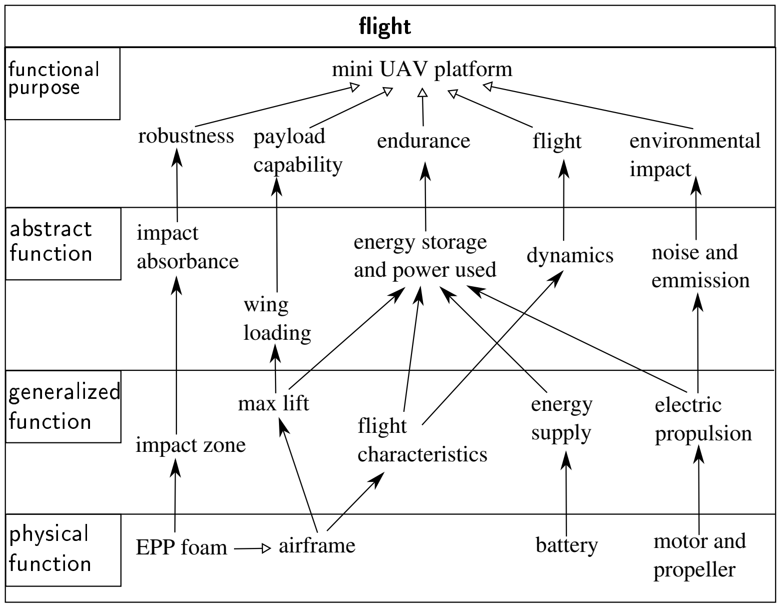


Figure 5.19: Graphical representation of the abstraction hierarchy at the flight level.

eficial to flight trials. The impact resistant EPP foam and Easystar's configuration with the motor mounted on the back, help survive rough landings. Environmental impact is a concern for acceptance of UAV flight testing with authorities and the community. The Easytar has electric propulsion that eliminates emissions and limits noise to a minimum.

Abstract function

The abstract functions for the four characteristics described on the 'functional purpose' level are as follows. 'Endurance' is achieved by the available energy storage and the average power usage. 'Dynamics' is achieved the aircraft's inherent stability and control response. 'Robustness' is achieved by the ability of the EPP foam to absorb relatively large impact energy before breaking. 'Environmental impact' is achieved by the low noise and zero emission.

Generalized function

The generalized function level represents generalized aircraft functions cut loose from their physical implementation. The four characteristics are defined by the co–function of different processes on the 'generalized' function level. Endurance in terms of parasitic drag coefficient, induced drag coefficient (depending on total weight) and propulsion efficiency. Handling qualities in terms of elevator / aileron effectiveness, excess power (climb performance) and drag (descent performance). Robustness in terms of allowable impact speed and attitude during landing. Environmental impact in terms of emissions and noise produced.

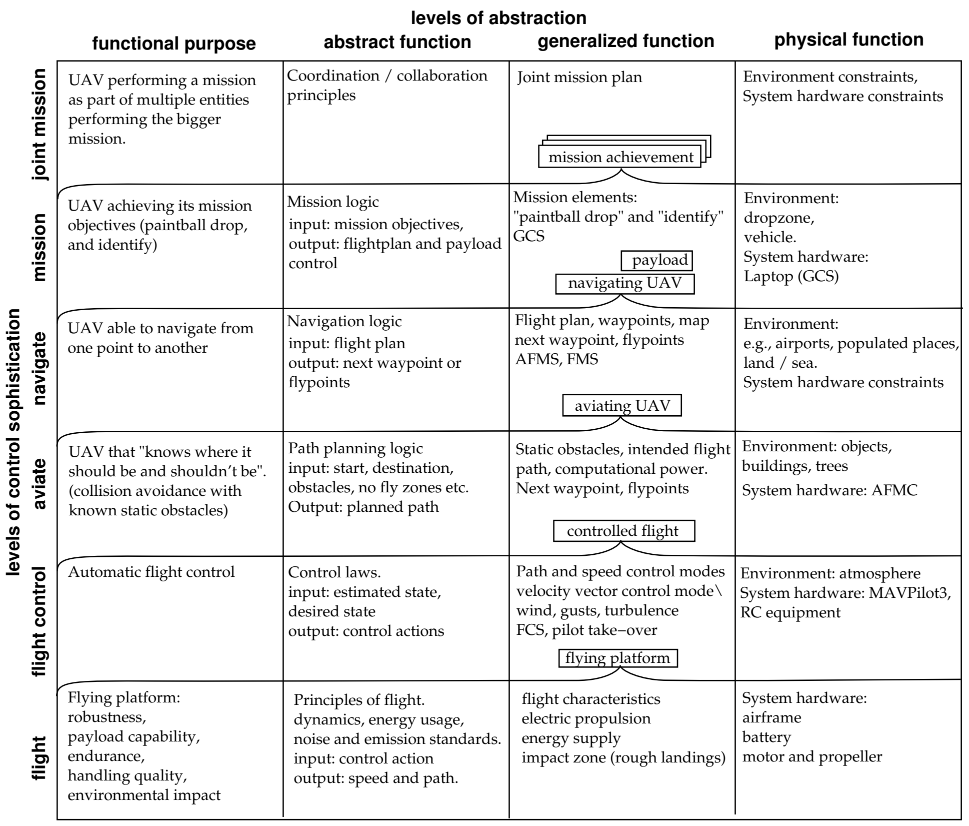
Physical function

The physical functions of the components playing a part in the four characteristics are represented. Endurance is expressed in the battery capacity, voltage and motor power consumption. Robustness of the Easystar is represented on this level as impact zone on the nose and belly. Environmental impact can be represented by noise production of the propeller and engine emissions (if not electric).

5.4.7 The abstraction–sophistication analysis

The individual levels of control sophistication have been discussed, together they form the abstraction–sophistication analysis. Here, we discuss how these come together in one representation and the relationship between the levels. Figure 5.20 shows an overview of the ASA. It show the same six levels of control sophistication and the levels of abstraction but the amount of detail has been reduced to prevent clutter. The nested structure is represented with the curly brackets in between the levels. They show that a level of control sophistication depends on the levels below it, and inherits the constraints that were introduced at that level. The functionality of the levels below a curly bracket is part of the control functions at the level above the curly bracket.

Each level of control sophistication becomes a single concept at the generalized function levels at the next, higher level of control sophistication. At the generalized function level, it is described independent of physical implementation. It provides a means to achieve the abstract functions of that level of control sophistication.



levels of control sophistication	functional purpose	abstract function	generalized function	physical function
joint mission	UAV performing a mission as part of multiple entities performing the bigger mission.	Coordination / collaboration principles	Joint mission plan	Environment constraints, System hardware constraints
mission	UAV achieving its mission objectives (paintball drop, and identify)	Mission logic input: mission objectives, output: flightplan and payload control	Mission elements: "paintball drop" and "identify" GCS	Environment: dropzone, vehicle. System hardware: Laptop (GCS)
navigate	UAV able to navigate from one point to another	Navigation logic input: flight plan output: next waypoint or flypoints	Flight plan, waypoints, map next waypoint, flypoints AFMS, FMS	Environment: e.g., airports, populated places, land / sea. System hardware constraints
aviate	UAV that "knows where it should be and shouldn't be". (collision avoidance with known static obstacles)	Path planning logic input: start, destination, obstacles, no fly zones etc. Output: planned path	Static obstacles, intended flight path, computational power. Next waypoint, flypoints	Environment: objects, buildings, trees System hardware: AFMC
flight control	Automatic flight control	Control laws. input: estimated state, desired state output: control actions	Path and speed control modes velocity vector control mode\ wind, gusts, turbulence FCS, pilot take–over	Environment: atmosphere System hardware: MAVPilot3, RC equipment
flight	Flying platform: robustness, payload capability, endurance, handling quality, environmental impact	Principles of flight. dynamics, energy usage, noise and emission standards. input: control action output: speed and path.	flight characteristics electric propulsion energy supply impact zone (rough landings)	System hardware: airframe battery motor and propeller

Figure 5.20: An overview of the abstraction–sophistication analysis for SmartUAV. The abstraction hierarchies for each level of control sophistication are combined into a single representation. The nested structure of the levels is shown by the curly braces: each level is based on the functionality of the level below and each level provides control function to the level above.

Moving up in level of control sophistication, the nature of the control problem changes from a fundamental vehicle control problem to more sophisticated[1] control of a complex system. The nature of the control problems at the lower levels of control sophistication, for example: flight and flight control, reflect mainly the vehicle characteristics and the environment. At higher levels of control sophistication, the nature of the control problems shifts to focus more on the bigger picture: navigation and mission control. Control at the navigate level is based on the existing principles of navigation: flight plans and waypoints. For mission control, the representation had to be developed utilizing mission elements as building blocks and logic to generate a flight plan from them. The representations for navigation and mission control are less guided by the physics of the world, and therefore, automation at the mission level is also less guided by physical constraint. Instead, their representation is based on invented solutions to achieve missions with the UAV system.

The ASA is used to represent the automation that was designed to solve the control problems. New functions that are provided by the automation introduce new constraints. The control solution at the flight control level sets constraints on the design of a solution at the navigation level. Control on the higher levels, be it manual or automatic, need to take the constraints into account at the lower levels of control sophistication. Therefore, the designed automation at one level of control sophistication is part of the work domain for automation design at the adjacent higher levels of control sophistication. The process of automation design is changing the work domain of the higher levels of control sophistication.

Constraint propagation

The ASA serves as a map to keep track of how constraints propagate through the designed system, which is particularly useful for the constraints that are introduced by design. During design, the ASA is an evolving map of the design landscape. An example of constraint propagation is discussed to illustrate how changes in the system can be mapped onto the ASA and track them though the system. The above presented discussion of the levels of control sophistication, give an overview of the system but they are not nearly detailed enough to represent the entire UAV system.

[1]From Merriam-Webster online dictionary: sophisticated means (i) deprived of native or original simplicity, (ii) highly complicated or developed, (iii) complex.

We take a particular problem that has been encountered during the design of the navigation functionality of the UAV system. The problem and its symptoms are illustrated by tracking the propagation of a change through the system in the ASA, shown in Figure 5.21. We start at the flight control level at the abstract function level, highlighted with the dotted ellipse. It represents the control laws and the gains in the control loops that need to be tuned to produce the desired control responses. At the flight control level the autopilot gains are chosen such that they produce accurate and stable flight.

We look again at reducing the control activity of the servos to reduce power consumption and wear, as discussed above for 'endurance' and 'durability' the flight control level. The gains are typically reduced, which also reduces the maneuvering rates of the UAV, and increases the maneuvering space needed. A larger turning radius results.

Note that the aviate level is skipped because it has not been implemented for the Easystar. At the navigate level of control sophistication, the concept of 'controlled flight' has changed. As shown by the means–ends link to 'flight plan interpretation' at the abstract function level, its properties determine *how* the flight plan interpretation is implemented. The larger turning radius due to reduced gains results in a mismatch between flight dynamics and the flight plan interpretation.

Figure 5.22 shows the resulting observed flight trajectory when two waypoints are placed too close to each other. The flight plan interpretation logic needs to detect when the UAV has passed a waypoint so it can command the flight controller to fly to the next waypoint. A common method to implement this is to define a distance d. When the distance between the UAV and the waypoint it is flying to is smaller than d, the navigation logic selects the next waypoint in the flight plan to fly to. In the illustrated example, the UAV will not successfully clear $WP4$ because the large turning radius will not allow the UAV to approach the waypoint closely enough. As a result the flight plan cannot be completed and the mission cannot be completed either.

In Figure 5.21, the crosses show where the problem becomes visible in the system. The path of constraint propagation shows that the problem has its roots in the Easystar platform (slow dynamics to start with), is aggravated by the gain settings of the flight control laws. The problem becomes apparent when the navigation logic does not seem to be able to cope with the large turning radius. This broken constraint propagates all the way op to the mission achievement.

The chosen solution was simple. The distance d in Figure 5.22 is increased to be larger than the largest expected turning radius of the Easystar. The Easystar can now clear each waypoint and complete the flight plan. It comes with a drawback, however. With a large d, the waypoint will be cleared before the UAV arrives at that point. Although solved at the navigation level, this altered constraint propagates up to the mission level: can the mission elements be completed successfully with this implementation of the navigation logic? In our case, the flight plans that were generated from the defined mission elements would still support the mission objectives. However, this example demonstrates the impact of subtle changes on the entire system complexity. In retrospect, a more elegant solution would have been one that stops the changes from propagating up. However, this will require a better analysis of what it means to clear a waypoint, as will be pointed out in Section 5.5.1.

5.4.8 Interface design

In the previous sections the main interface features for controlling the UAV from the GCS has already been discussed. It has been found useful to organize the interface around the levels of control sophistication. At the GCS, the operator is able to choose the level of control at which he want to control the system. Figure 5.23 gives a screenshot of SmartUAV running in a typical simulation configuration. At the bottom of the mission view window, four tabs are shown: 'Aviate', 'Navigate', 'Mission' and 'Planning', with the 'Navigation' tab selected. The first three tabs coincide with the levels of control sophistication.

By selecting a tab, the interface is configured to support control at the selected level of control. By selecting 'Mission', the map presents symbols for the mission elements and the interface shows a list of mission elements that can be selected and manipulated (see also Figure 5.9). The UAV can be commanded to proceed with the selected mission elements at any time. The map symbology at the mission level is the least cluttered and allows the operator to focus on the mission objectives. This view does, however, not explain the predicted flightpath.

When the operator wants to see how the predicted flightpath resulted, he can select the 'Navigate' tab. The map symbology at the navigate level shows the complete flight plan and the waypoints that shape the (predicted)

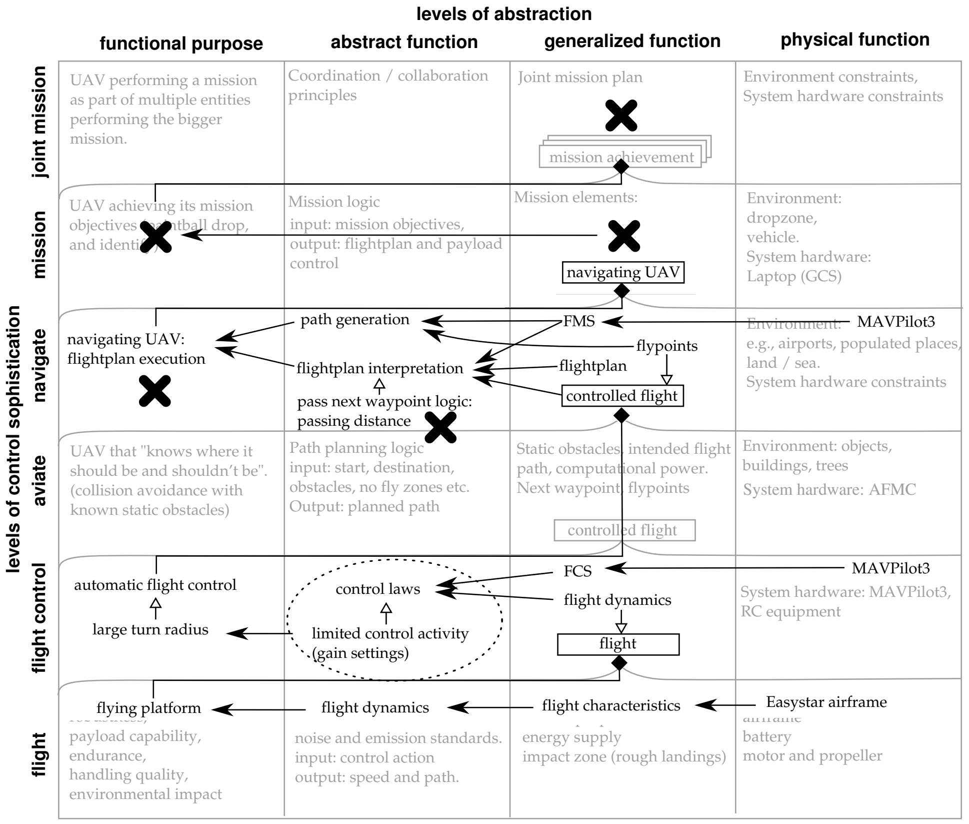

levels of abstraction
functional purpose
abstract function
generalized function
physical function
levels of control sophistication
joint mission
mission
navigate
aviate
flight control
flight
UAV performing a mission as part of multiple entities performing the bigger mission.
Coordination / collaboration principles
Joint mission plan
mission achievement
Environment constraints, System hardware constraints
UAV achieving its mission objectives paintball drop, and identification
Mission logic input: mission objectives, output: flightplan and payload control
Mission elements:
navigating UAV
Environment: dropzone, vehicle. System hardware: Laptop (GCS)
navigating UAV: flightplan execution
path generation
flightplan interpretation
pass next waypoint logic: passing distance
FMS
flypoints
flightplan
controlled flight
MAVPilot3
e.g., airports, populated places, land / sea. System hardware constraints
UAV that "knows where it should be and shouldn't be". (collision avoidance with known static obstacles)
Path planning logic input: start, destination, obstacles, no fly zones etc. Output: planned path
Static obstacles, intended flight path, computational power. Next waypoint, flypoints
controlled flight
Environment: objects, buildings, trees
System hardware: AFMC
automatic flight control
large turn radius
control laws
limited control activity (gain settings)
FCS
flight dynamics
flight
MAVPilot3
System hardware: MAVPilot3, RC equipment
flying platform
flight dynamics
flight characteristics
Easystar airframe
payload capability, endurance, handling quality, environmental impact
noise and emission standards. input: control action output: speed and path.
energy supply impact zone (rough landings)
battery motor and propeller

Figure 5.21: Constraint propagation mapped onto the ASA (shown in Figure 5.20).

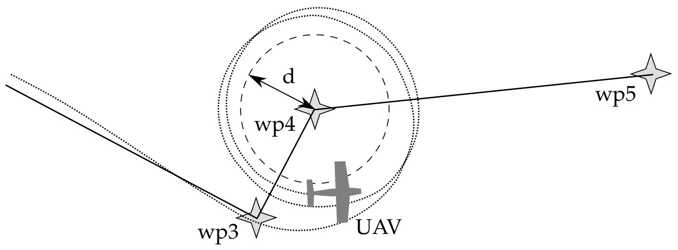
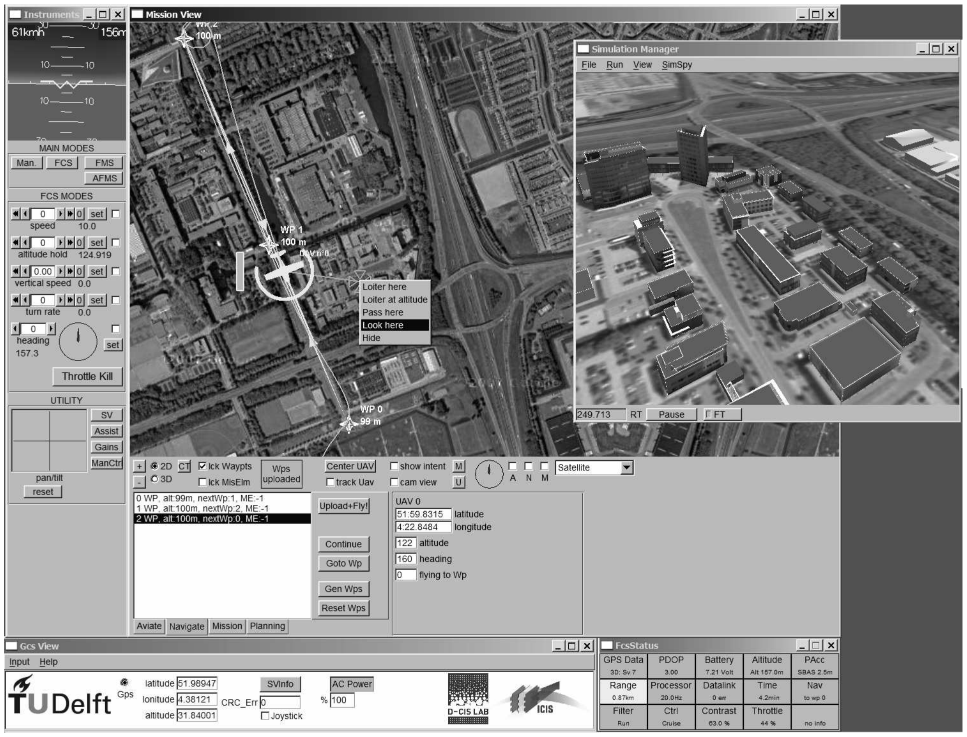


Figure 5.22: A probable trajectory when a design mismatch between the UAV's flightplan logic and the flight dynamics is created. The UAV is unable to approach the waypoint (WP4) close enough (distance *d*) to clear it and trigger the navigation logic to command the autopilot to the next waypoint (WP5).



speed 0, 10.0
altitude hold 0, 124.919
vertical speed 0.00, 0.0
turn rate 0, 0.0
heading 0, 157.3

249.713

0 WP, alt:99m, nextWp:1, ME:-1
1 WP, alt:100m, nextWp:2, ME:-1
2 WP, alt:100m, nextWp:0, ME:-1

UAV 0
51:59.8315 latitude
4:22.8484 longitude
122 altitude
160 heading
0 flying to Wp

latitude 51.98947
lonitude 4.38121
altitude 31.84001
CRC_Err 0
% 100

GPS Data 3D: Sv 7	PDOP 3.00	Battery 7.21 Volt	Altitude Alt 157.0m	PAcc SBAS 2.5m
Range 0.87km	Processor 20.0Hz	Datalink 0 err	Time 4.2min	Nav to wp 0
Filter Run	Ctrl Cruise	Contrast 63.0 %	Throttle 44 %	no info

Figure 5.23: SmartUAV's main GCS interface showing the instruments panel, the map view, simulation manager (providing a simulated camera view), the UAV status panel, and the GCS status panel.

flightpath, as was shown in Figure 5.11, The interface presents a list of the waypoints and their attributes to be changed by the operator. The operator can direct the UAV to any waypoint by selecting it and commanding the UAV to fly there.

By selecting the 'Aviate' tab, the operator is supported to interact with the system in terms of aviate constraints in the case of the helicopter simulation as discussed above. The 'Planning' tab provides a small interface for saving and loading missions and flight plans. The flight level does not use a map representation and is supported through the 'Instruments' window on the left of Figure 5.23.

To support the operator (and design engineers) in understanding the instantaneous actions of the UAV, two symbols are shown at all times. First, a line is drawn to the *next waypoint* that the UAV is flying to, revealing the goal of the FMS. Second, the *flypoint* that the UAV is flying to is shown as an orange triangle, revealing the goal of the FCS.

5.5 Discussion

WDA was performed throughout the design of SmartUAV. Both the design and the analysis grow at the same time, like scaffolding rises with a new high–rise building. The scaffolding needs to be in place for the building to rise and at the same time the building needs to be there for the scaffolding to be supported. After construction the scaffolding is removed and the building is left as the product of the construction process. Likewise, SmartUAV is the result of a process where the analysis and design grew together.

Because WDA for EAD was developed during the design of SmartUAV, only a limited part of SmartUAV could properly benefit from the generated insights. Furthermore, SmartUAV served other purposes than this theoretical investigation. The engineering interests and deadlines for demonstrations often forced an implementation that lacked the kind of elegance that we sought through EAD. The number of people that have worked on the project (over ten were directly involved) and the sheer size of the project (over two thousand source code files, excluding external libraries) prevented the analyst to look closely at the implementation. As a result, the degree to which the design has improved due to EAD principles development and application cannot be quantified. Taking this into account, the following insights were obtained.

5.5.1 The analysis

Levels of control sophistication

The purpose of the UAV system is to perform missions. This reflects a very broad view on the system and therefore the analysis includes more abstract control problems than the domains discussed in Chapters 3 and 4. As control over the UAV is made more sophisticated, from basic flight toward mission functionality, more constraints are introduced by the environment and automated processes. During the design and application of the theory, it became clear that WDA would benefit from making this nested structure explicit in the analysis framework. By defining the levels of control sophistication, each control problem is represented and the constraints specific to each level are represented in the analysis.

Specifically addressing the nested structure during analysis helped to modularize the subsystems of the UAV. For example, the definition of the aviate level led to the definition of the *flypoints* well before any implementation of the aviate level was defined. Their definition avoided direct interpretation of waypoint interpretation by the FCS based on the principle structure of the levels of control sophistication. The *flypoints* provide a way to stream real–time generated path points to the FCS, allowing any interpretation of flight plan or mission to be made in the future. The aviate level of control sophistication became a placeholder for their definition and the technology that would use them, avoiding a patchwork approach when the aviate levels was implemented without this placeholder.

When performing WDA, humans and actions should not be included in the WDA, as stated by Rasmussen et al. (1994) and Vicente (1999). We made the parallel between actions performed by the automation and actions performed by a human actor. By following the stated rule, the actions that the automation performs should not be represented in the analysis. However, automation changes a work domain. These changes impact how the system is used by human operator and further automation. They set new constraints in the work domain and need to be taken into account in the WDA. The introduction of the levels of control sophistication allow the inclusion of designed automation in WDA.

The constraints introduced by automation at one level of control become part of the work domain of the next higher level of control sophistication. This incremental approach to the analysis and automation design allowed the designer to take the automation introduced constraints into account in the analysis. Each level builds on the functions and constraints of the levels below, achieving more sophisticated control.

The hierarchy of the levels of control sophistication represent a set of *nested work domains*. Before introducing the automation at the 'aviate level', the levels below it for the work domain of the aviate automation. In turn the introduced automation at the 'aviate level' is part of the work domain of the next higher level, the 'navigate level'. This incremental design approach allows the (incremental) inclusion of automated functions in the WDA. Additionally, the levels of control sophistication span the work domain from the basic flight level to the joint mission level. By separating the different control problems that require different representations, the levels of control sophistication allowed a larger system scope without attenuating the depth of the analysis.

Abstraction–sophistication analysis

At each level of control sophistication an abstraction hierarchy is applied to further analyze the control problem specific to that level. The control problems also change in their nature. The lower levels of control sophistication are more guided by the physics and physical elements of the world, while the higher levels are more guided by artificial and abstract concepts used to solve the control problems. The analysis of the 'flight level' is guided by physical laws, such as the law of conservation of energy. The analysis of the 'flight control level' is guided by the designed control laws, the control modes that the autopilot supplies, and the ability to deal with wind, gusts, and turbulence. The concepts at this level are still closely linked to the physics of flight and the path in the physical world. At the 'aviate level', the analysis is guided by the pathfinding problem and by the constraints that are introduced by the supplied solution.

Going further up in levels of control sophistication, at the 'navigation level' the analysis focuses on the artificial concepts of flight plans and waypoints. At the mission level the analysis revolves around the artificial definition of mission elements that were invented to be able to achieve the I/EMAV missions (EMAV09, 2009).Abstraction also takes place along the control sophistication dimension of the ASA, from concrete at the lower levels to abstract at the higher levels.

Reviewing the abstraction–sophistication analysis

The ASA has a strong focus on the added automated functions. The analysis would benefit from more explicit representation of the nested work domains that results from the automation that is added at each level of control sophistication. In future analysis, each level of control sophistication is recommended to be represented in two steps. The first step describes the work domain independently of the new functions to be added. The second involves the design of the automation to fit the work domain constraints.

The analysis of the manual longitudinal flight control task and TECS in the previous chapters illustrated the two steps and the fit between work domain and the automation. The aircraft was the work domain with the energy based control coupling that was represented by the reservoir analogy. The

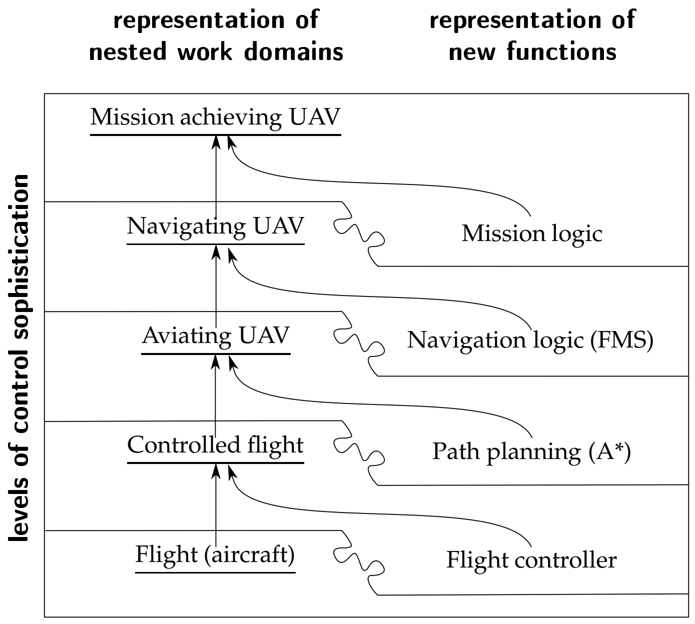


Figure 5.24: For each level of control sophistication the representation of the work domain constraints independent of the automated functions is the first step (left column). Then the new functions can be designed to fit those constraints. Together they form the new work domain of the next higher level of control sophistication.

fitted automation was the energy based control decoupling of TECS that is represented by the pushrods analogy. The analogies fit together and form a control solution as shown in Figure 4.6.

The UAV system analysis of this chapter misses the mark with respect to representing the work domain and the control solution separately. To clarify, before designing the A* path planning logic into the AFMS, the work domain made up by the 'controlled flight' functions should have been detailed further. This way, the constraints that are introduced by the 'controlled flight' would have been made more explicit prior to design of the aviate logic. Similarly, the navigation capabilities of the UAV should be represented in WDA prior to adding the mission logic at the mission level. The two–step process per level and the fit between work domain and the designed automation is illustrated in Figure 5.24.

5.5.2 Reflecting on interface design

The ASA supports EID in the same way that the abstraction hierarchy is a framework for representing the information content and structure in the interface. According to EID, the interface should represent the information at each level of abstraction. In addition, when multiple levels of control sophistication are identified, the interface should support the operator to view the level(s) of control sophistication that he is interested in.

5.5.3 Reflecting on Ecological Automation Design

EAD has been introduced in Chapter 1 by using the concept of the shared domain representation, see Figures 1.1, 1.2, and 1.4. The concept of the shared domain representation illustrates that the human operator and the automation should work with the same constraints. Working with the levels of control sophistication has caused the structure of the design problem to become nested as well. Figure 5.25 shows that the structure of Figure 1.4 is applied at each level of control sophistication. Each level of control sophistication is shown to be the work domain of its adjacent higher level of control sophistication. This structure reveals that the proper functioning of each level relies on the functioning of automation and human control at each lower level of control sophistication. It makes explicit that automated functions can rely on functions that are provided by the human operator. This is also illustrated by the *car navigation system–driver* dependency that is exemplified in Appendix A.

Automation introduced complexity

How well *automation induced complexity* has been limited by EAD is difficult to quantify in the study of this chapter. The focus has shifted from limiting complexity in the designed automation to keeping track of the constraints that are introduced through (automation) design. This is the first step in any attempt to limit them, but a claim that the application of EAD has limited the complexity in SmartUAV cannot be substantiated. However, as illustrated by the constraint propagation example, in Section 5.4.7, the ASA allows keeping track of the impact of subtle changes at the lower levels of control sophistication, impacting functioning at the higher levels of control sophistication.

Future development efforts can focus on stopping this type of constraint

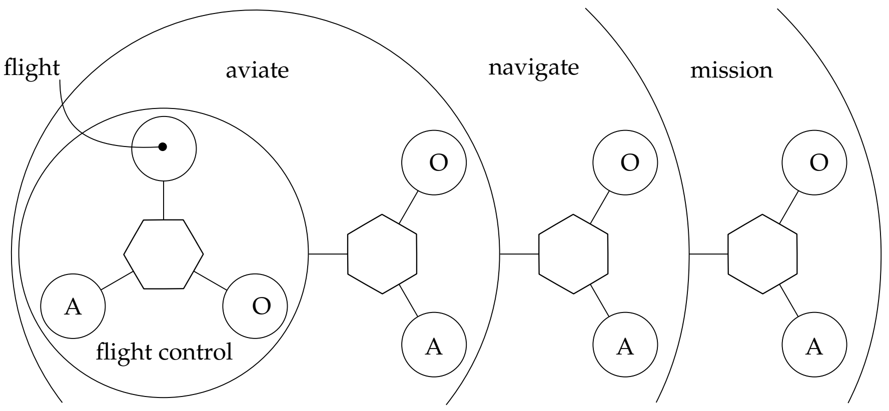


Figure 5.25: Reflecting on the structure of the ecological automation design problem. Each level of control sophistication forms the work domain of the higher levels of control sophistication. (Legend O: operator, A: automation.)

propagation without adding complex overhead control logic. Improving WDA, taking into account what was outlined in Section 5.5.1, will enable better definition of the interfaces between the levels of control sophistication and the fit between automation and its work domain.

The most powerful tool for reducing apparent complexity is choosing the representation of the constraints that the automation works with. To exemplify, we look at the A* algorithm implementation for obstacle avoidance. In this case it is unavoidable that automation works with a different set of constraints than the human operator. This is mainly due to the restriction in computational power and available algorithms. The representation that the A* algorithm works with is a cost map which is a grid that approximates the world with a certain, lower, resolution. Additionally, freedom of movement in this grid is restricted to 8 directions of movement in the horizontal plane. These limitations are mainly necessary to limit the required processing time to come to a solution. A translation from how a person sees obstacles (buildings) to how the automation sees them on the cost map has to be made. Proper representation can assist in that, but this automated solution inevitably introduces new constraints in the system. Through the interface of the *aviate level* the operator could manipulate the aviate constraints, directly manipulating the cost map of the A* algorithm. These constraints are the natural language of the aviate level and are used to control the behavior of the algorithm without altering the algorithm itself. This satisfies the concept of the *shared domain representation* at the aviate level.

6 Case study: The sailboat racing domain

In the previous three chapters, the principles of Work Domain Analysis (WDA) have been demonstrated for the design of a graphical interface, the analysis of automated flight control, and the design of the control logic of a mini–UAV system. To be able to generalize the approach to WDA, this chapter shows the analysis of another vehicle control domain: sail boat racing.

In this domain, the equipment, rules and way of working have co–evolved over time and have resulted in well described domain knowledge and *best practices*. The purpose and goals of the boats are the same: win the race. Furthermore, in contrast to the vehicle control domains that were the topic of Chapters 4 and 5, this domain lacks automation. Therefore, the focus will be on the representations for control instead of the actual control systems as was the case in Chapters 4 and 5.

The goal of winning the race is achieved by both mental and physical human teamwork. This means that the human crew is involved in more than decision making and supervisory control. Their physical and mental performance is part of the proper functioning of the sailboat as a system. These functions are part of the work domain and will be included in the WDA through a level of control sophistication.

Most readers will not be very familiar with this work domain, thus it also lends itself well as an example for capturing and conveying work domain knowledge. To limit the space and detail required for this chapter, only the windward leg of a race is considered. First, an introduction of the selected subset of the sailing domain is given. Then, the choice of levels of control sophistication is presented. This is followed by the application the abstraction hierarchy to each level of control sophistication. Finally, constraint propagation is illustrated with the ASA.

6.1 Introduction

The domain expertise used throughout this chapter comes from the author's racing experience and from Bill Gladstone's books on performance racing (Gladstone, 2000, 2006). Most of the author's racing experience comes from sailing the IJspegel trophy on the North Sea off the coast of Scheveningen. This is a series of 20 races that takes place every year spread out over the winter months. Each race is usually two laps around a course consisting of an upwind and a downwind leg. The distance between the windward mark and leeward mark depends on wind and weather conditions but is normally around 1.2 nm. Figure 6.1 shows the typical short course of the IJspegel races. Weather permitting, two races are held on one day. The boats that were sailed on are an X–Yacht 372 and a Baltic 45 with a crew of 7 to 9 people.

6.1.1 The racing pyramid

Gladstone (2000, 2006) presents the *racing pyramid* as a view on the ingredients of successful sailboat racing. The racing pyramid is a hierarchy of three levels, see Figure 6.2. The base level is labeled *boat handling*, a crew must be able to handle a boat before they can successfully race it. The middle level is labeled *boat speed*. Speed is the essential ingredient of almost any race. A good understanding of how to achieve best speed is essential because rivals usually don't sail alongside and speed differences are difficult to see. The top of the pyramid is labeled *tactics*. Tactics can give a crew a competitive edge over rivals with similar boat speed and similar boat handling. To win, a crew must work its way up the racing pyramid: boat handling must be second nature, and boat speed must be at its best. Without those two ingredients, a crew with good tactics alone is unlikely to win a race.

Boat handling involves the constant physical action needed to bring the boat and equipment in the state that enables the current and next manoeuvre at the best speed. It is a main challenge for the crew met in every minute of

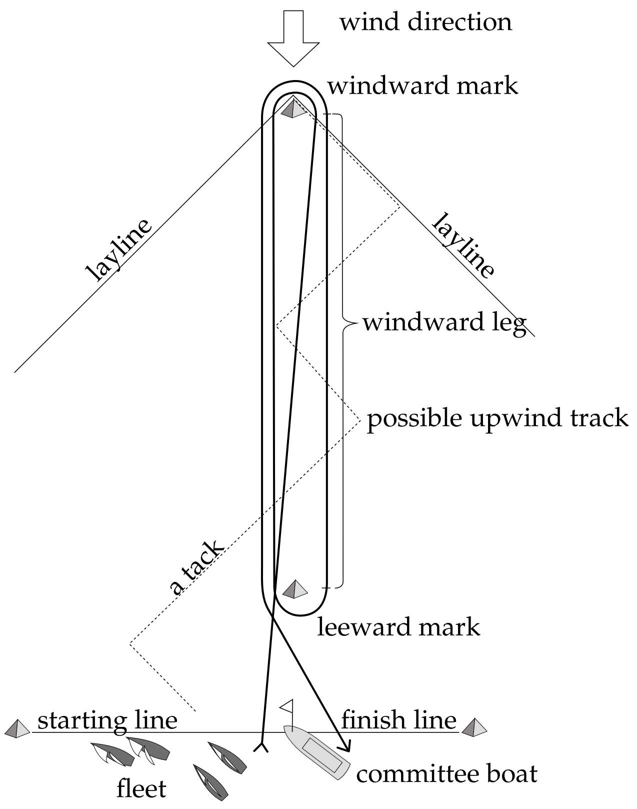


Figure 6.1: The IJspegel trophy's short course consisting of an windward leg and a leeward leg. Sailing the windward leg is the topic of the analysis. Sailing boats cannot sail straight into the wind and must tack to reach the windward mark.

the race. Boat handling involves a large variety of physical actions that need to be executed timely and accurately. Usually wavy and windy conditions disturb moving, grabbing, pulling, cleating, and other actions that would otherwise be easily executed. Crew members have to coordinate actions in a timely manner to execute a manoeuvre properly, their success is interdependent. If one crew member struggles, failing to release a line, the rest could find themselves in the awkward position of trying to drag a $180m^2$ spinnaker out of the sea, not to mention the impact this will have on boat speed and the overall race.

Boat speed is essential for winning a race. Every moment of the race the crew monitors speed and makes small adjustments to increase it. These adjust-

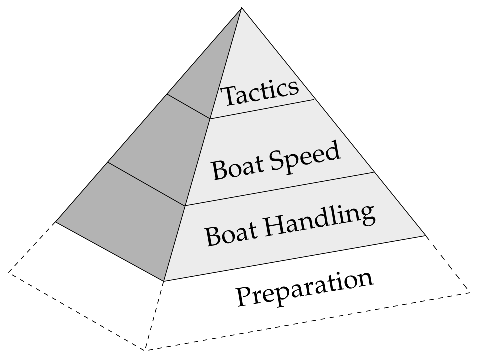


Figure 6.2: The racing pyramid has three levels of competencies that a crew must master to win a race. *Preparation* is its foundation (Adapted from Gladstone(2006)).

ments include continuous sail trim and steering with respect to the wind, waves, marks, and rivals. Sail trim can be hard work and it is very frustrating for a trimmer to work hard and achieve excellent boat speed on a leg, only to lose one or two positions when a manoeuvre is wrongly executed because the crew has not mastered boat handling yet. For maintaining a high average boat speed, it is essential that the crew has excellent boat handling skills as pointed out by Gladstone (2000, 2006).

Tactics builds on the crew's ability to handle the boat and maintain a high boat speed. Without a team that handles the boat well and achieve best speed, the tactician is uncertain of the ability to pursue tactical advantages. For example, a crew that has a bad reputation for boat handling and/or for building speed may be better off not tacking much. This leaves the tactician weighing the risk of a badly executed tack versus taking advantage of a tactical situation. The affordances of tactics are often more readily seen and understood by experienced crew members that master handling and boat speed.

The hierarchy of the racing pyramid (Figure 6.2) illustrates the nested structure of control problems in the work domain. For the boat to perform the crew has to perform, and the crew becomes part of the work domain in terms of the functions they provide to higher levels of the racing pyramid. For example, a tactical decision can take the performance of the crew into account. This is the same nested dependency of the levels of control sophistication introduced in Chapter 5. The levels of the racing pyramid can be seen as levels of control sophistication, as will be discussed later in this chapter.

6.1.2 The windward leg

The scope of this chapter's WDA is the windward leg. As shown in Figure 6.1, the windward leg is the part of the course where the next mark is straight into the wind. This is particularly challenging because sailboats cannot sail straight into the wind. In order to reach the windward mark the boats need to tack by sailing on starboard and port bow subsequently as shown by the possible track in Figure 6.1. Most racing boats achieve an angle with respect to the true wind angle of approximately 45° when sailing *close haul* [1]. The velocity component into the wind is the velocity made good (VMG), which is the main measure for speed performance. The precise angle that a boat can achieve differs per boat and weather conditions, but for convenience of explanation the figures of this chapter show boats sailing close haul at a 45° angle with respect to the true wind. Consequently, they are shown tacking through an angle of 90°.

Figure 6.3 shows two boats sailing to the windward mark. For illustration, boats A and B sail at the same speed and angle to the wind but have chosen different paths to the windward mark. At each of the four positions shown they are on the same line of equal position (LEP), meaning that neither is ahead of the other. In perfect homogeneous conditions the exact track sailed does not matter as long as the boats sail close haul and do not sail past the laylines.

The laylines are the approach lines for the windward mark, these are also shown at 45° with the true wind. A boat has sailed too far when it sails beyond the laylines, or out of the square (or rectangle) that is spanned by the laylines and the leeward mark. Therefore the square of Figure 6.3 is the part of the sailing course that is of interest for the windward leg.

Sailing conditions are almost never homogeneous, and in practice tactical gains often outweigh the small speed penalty of a well–executed tack. Deciding where to sail and when to tack is typical of strategy and tactics discussed below.

[1]Close haul is the mode of sailing on the windward leg to achieve the best speed into the wind.

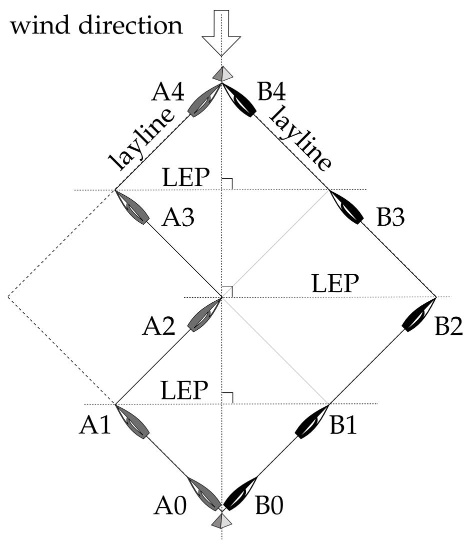


Figure 6.3: Upwind sailing towards the windward mark in perfect homogeneous conditions. When both boats sail with the same speed, they will constantly be on the same Line of Equal Position (LEP) and reach the windward mark at the same time, regardless of when or where they tack in the square.

6.1.3 Instrumentation

Instrumentation provides information for feedback that is essential for goal–directed actions. Information that cannot be perceived directly or accurately is typically presented by onboard instrumentation. Examples of directly measured variables are: boat speed (measured by water flow), speed over ground (typically measured by GPS), wind direction, and wind speed. From this data more meaningful variables can be computed like the VMG to allow the helmsman to find the optimal heading to sail as fast as possible into the wind.

Not all instrumentation is electronic. Telltales are strings attached to the sails near the leading edge of the genoa and the leech of the mainsail. They provide information about the airflow around the sails and facilitate trimming and steering. These are similar to the first aeronautical instrument used by the Wright Brothers to visualize the Wright Flyer's slip angle and angle of attack (Crouch, 1989).

A wind graph is another form of instrumentation (Gladstone, 2006). It is a table of encountered wind directions noted down by the tactician. With it he has an overview of how the wind direction is developing. It helps him to anticipate and identify wind shifts in order to take advantage of the tactical opportunity they provide.

6.1.4 The crew

All actions are executed by the crew. Their presence is essential to make the system work. A good, regular crew is fundamental to good boat handling, which in turn is a prerequisite for successful racing. Gladstone (2000) describes crew organization principles designed to make a crew do their work as well as possible. These, and the performance of the crew are part of the work domain analysis. However, crew members can be seasick, tired, or distracted which may all interfere with their performance. These physiological, psychological, and social properties are not part of the work domain analysis of this chapter.

6.2 Abstraction–sophistication analysis

The goal of this analysis is to demonstrate the ASA applied to a vehicle control problem in a fresh work domain, and not to teach sailboat racing. Completing the entire WDA for this domain would be similar to writing a comprehensive book on racing sailboats. In addition, trying to describe the entire work domain in limited space will result in fairly general terms and loose coupling between them. By choosing a subset of a domain, the limited scope will allow sharp descriptions with a tight coupling.

At each level of control sophistication only a small number of constraints is taken into account to illustrate the ASA. To keep the example crisp and the chapter brief, not the most difficult and intriguing examples are used, but basic ones that offer insight into how they work across all the levels of the analysis. The available domain knowledge is discussed according to the levels of the ASA.

A top–down approach is chosen for the explanation of this work domain. It will give the reader a better appreciation of the involved strategy and tactical game of sailboat racing. The skill needed for working with the constraints that sail boats impose on winning the game makes sailboat racing a challenging and fun sport.

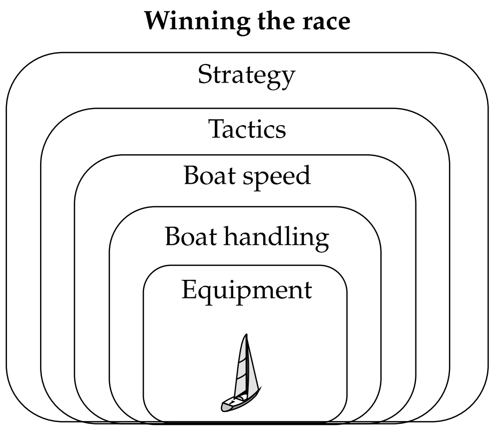


Figure 6.4: The levels of control sophistication are similar to the racing pyramid but with 'strategy' and 'tactics' as different levels, and with equipment as the lowest level.

6.2.1 Levels of control sophistication

The first step is to identify the levels of control sophistication. The levels of control sophistication *guide* abstraction into a direction meaningful to the control problem of that specific level. The levels are similar to the levels of racing pyramid, with some differences. Figure 6.4 shows the levels of control sophistication.

Top–down, the first two levels are *strategy* and *tactics*. In contrast to Gladstone's racing pyramid, these are two separate levels. Both *strategy* and *tactics* are typically dealt with by the tactician, but they represent different sets of constraints. **Strategy** is the top level and deals with the overall view of the race course and with making the sailing plan before sailing the race. A strategy can be updated during the race to accommodate changing (weather) conditions but the focus remains on the bigger picture. At the **tactics level**, the strategy is implemented. It deals with decisions–making during the race on a smaller spatial scale and shorter time scale. The strategy tells the tactician where he wants to go, and he uses tactics to go there. This includes dealing with wind shifts (local weather conditions) and applying the racing rules to deal with rivals. The third and fourth levels are *speed* and *handling*. The **speed level** deals with making the boat sail fast towards the windward mark. The **handling level** deals with the physical actions of the crew operating the boat. The fifth and bottom level is the **equipment** level. Boat equipment must be in a competitive condition to race successfully. This level is similar to *preparation*, what Gladstone calls the racing pyramid's hidden foundation. A crew has to be well prepared as well, their base competence belongs to the *preparation level* while their performance during the race, under the governing conditions belongs to the *boat handling* level.

Each level of control sophistication relies on the levels below it. Without the ability to apply tactics, the execution of a strategy will be weak. Without the ability to sail fast, tactics won't give an advantage. Without proper boat handling, speed cannot be achieved. Next, the control problems found at the levels of control sophistication are detailed, and the abstraction hierarchy is applied to map out the structure of the control problems.

6.2.2 Strategy level

Strategy focuses on the big picture, the racing plan. The racing plan is based on local knowledge and predictions of wind, wind shifts, and current. The goal is to make a plan that takes advantage of the local sailing conditions.

The square that spans the racing area of interest (Figure 6.3) is shown again in Figure 6.5. This time the conditions are not homogeneous; winds are stronger and there is a weaker current to sail against on the right hand side. Therefore, the right hand side is the favored side of the windward leg. The corners should be avoided due to tactical considerations because the impact of windshifts is larger towards the sides as explained in Section 6.2.3. In this example a good strategy is to sail to the right of the middle as boat A does. In fact a rule of thumb on the strategy level states: *Sail in the middle to the favored side.* Local knowledge of the sailing environment (winds predictions, currents) is a prerequisite for determining a good strategy. This is illustrated by Figure 6.6(a) that shows how a shoreline tends to cause a persistent wind shift. Understanding this situation will result in the strategy sailed by the boat A instead of that of boat B. Boat A has planned the longest tack after the onset of the persistent shift which allows a shorter distance sailed to the windward mark.

The rule of thumb, at the strategy level, for dealing with persistent wind shifts is: *Sail to the new wind.* Sail the headed tack[2] first before it deterio-

[2]A windshift is experienced as a header when it turns the bow away from the windward mark, sailing close haul.

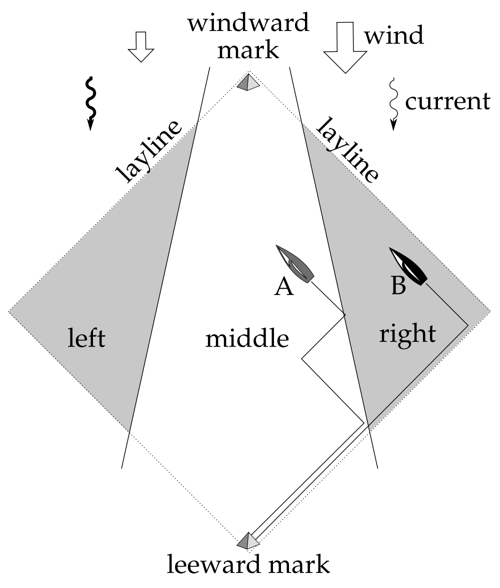


Figure 6.5: In this example the right side has a strategic advantage due to stronger winds and weaker currents, and is the favored side. The middle is less sensitive to wind shifts and has a tactical advantage. Boat A sails a good strategy.

rates too badly and then sail the lifted tack[3] as it has become best. Boat A in Figure 6.6 adopts this strategy.

Oscillating wind shifts are dealt with in the opposite way at the tactics level. Because oscillating winds shift back and forth, it is best to take advantage of the lifted tack immediately. Understanding the conditions, thus knowing when to anticipate persistent shifts and when to anticipate oscillating wind shifts is of strategic importance for the racing plan.

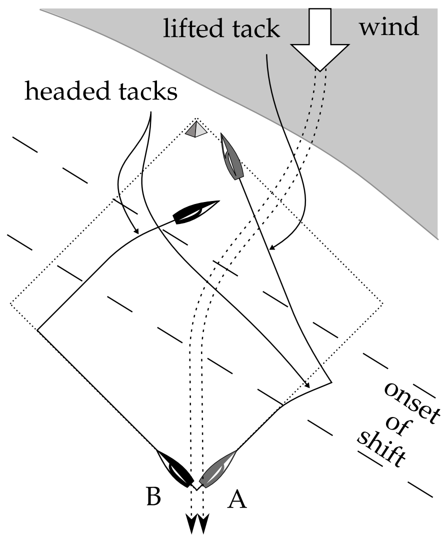
The abstraction hierarchy

Figure 6.7 shows the graphical representation of the abstraction hierarchy for strategy. The levels are discussed below.

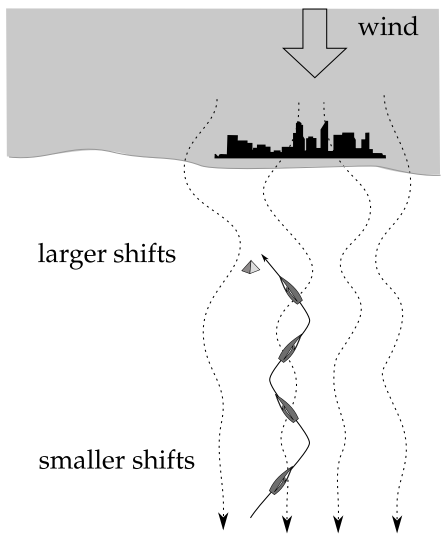
Functional purpose level

The functional purpose of the strategy is to find the most favorable conditions for sailing based on knowledge and predictions of the environment.

[3]A windshift is experienced as lifter when it turns the bow toward the windward mark, sailing close haul.



(a) Persistent wind shift: offshore winds are locally shifted towards the perpendicular of the shore. Two boats are shown: boat A taking advantage of the shift and boat B ignoring the shift.



(b) Oscillating wind shifts: a city or hill can create oscillating wind shifts. Oscillating wind shifts have to be dealt with on the tactics level.

Figure 6.6: Two types of wind shifts are identified: persistent wind shifts and oscillating wind shifts.

For sailing the windward leg, although a bit simplified, this comes down to finding the favored side of the square. If all conditions during the race could be known beforehand, the perfect plan could be made. Even when the weather predictions are accurate, in reality the exact local weather conditions will vary and will cause variations on the strategy. However, this is the domain of tactics that is discussed below.

Abstract function level

To formulate the racing plan, thus to find the favored side, the available information about the sailing conditions is processed. This information flow and the organization of the knowledge is typical of the abstract function level. The knowledge of (predicted) weather, sea currents, and boat properties are combined to form the sailing plan. The tactician (who is also in charge of strategy) will ask these questions to form a mental image of the conditions to find the advantages. How is the racing square positioned with respect to the wind, is there an advantage? On which side can the best VMG be expected based on wind and current? Are wind shifts expected, are they persistent or oscillating, and on which bow to sail when they occur? The combination of these factors before and during the race, requires the skill of weighing the influences and estimate where the net advantage will be the largest.

Generalized function level
The elements that are used to reason with at the abstract function level are represented at this level. These are the square of the windward leg (shown in Figure 6.3), the power provided by the wind, the influence of the sea currents, and the persistent wind shift due to the presence of a shore line and wind direction. The *ability to apply* tactics during the race is also represented at this level as a function that is provided by the tactics level. The latter builds on all lower levels of control sophistication. For example, when a team is exhausted and weak, ability to apply tactics changes. The strategy may have to changes as well.

Physical function level
The physical variables defining the system state at this level and in this example are the wind, sea current, shore line, and the location of the windward mark.

6.2.3 Tactics level

Tactics serve to sail the best strategy and adapt to changing circumstances. Strategy and tactics are tightly coupled but tactics mainly deals with constraints that the strategy cannot foresee. Racing sailboats is about finishing first and, therefore, the position relative to the rivals is the main criterion for success.

Here we look at only two of the many mechanisms useful to tactics. The first is an example of dealing with the local sailing environment: oscillating wind shifts. The second is an example of dealing with rivals by applying racing rules. Both represent constraints and opportunities for sailing the planned strategy faster than rivals.

Oscillating wind shifts

Winds are always shifting. The impact of wind shifts is considerable and it is up to the tactician to deal with them. The relative position of boats in the

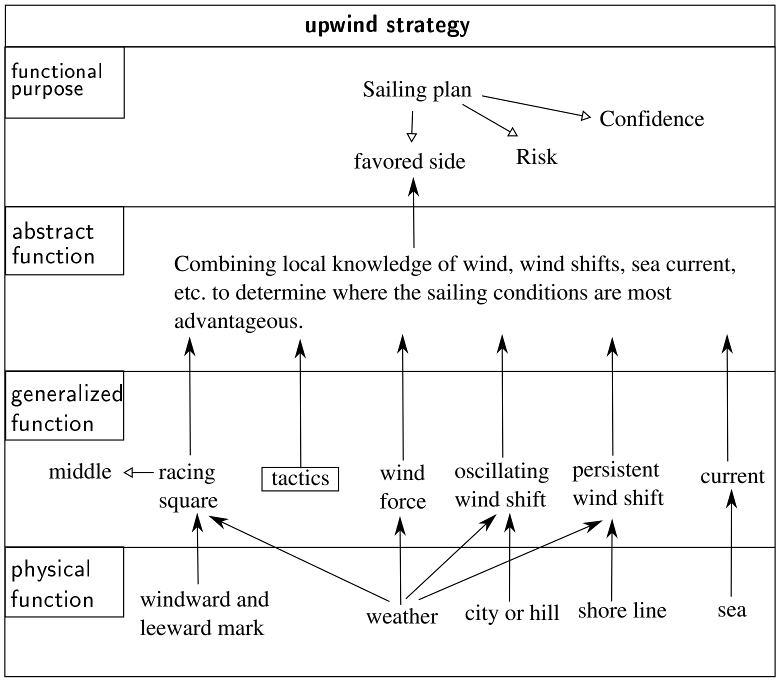


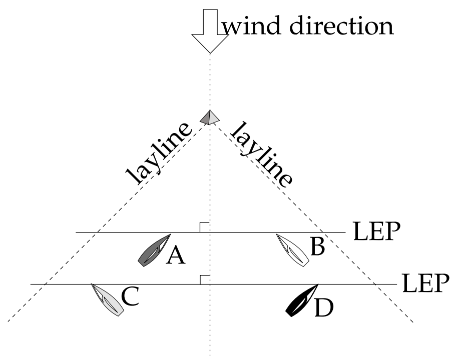
Figure 6.7: Graphical representation of the abstraction hierarchy at the strategy level of control sophistication.

fleet are measured from the windward mark in the direction of the wind. Consequently, when the wind shifts the relative positions also shift.

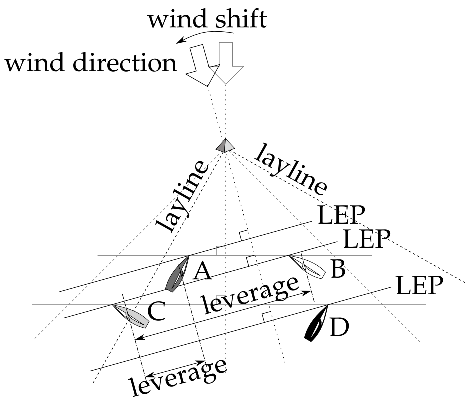
Figure 6.8(a) shows a situation of a fleet of four boats sailing to the windward mark. Boats A and B are on the same line of equal position (LEP) and share the first position. Boats C and D are also on the same LEP and share the second position. As the wind shifts, the LEPs rotate as shown in Figure 6.8(b). Boat A now leads in first place, boats B and C share second position and boat D is last.

Due to the shift, boat C gained twice as much on boat B (a whole position) than on boat A. The reason is that boat C's *leverage* with respect to boat B is larger than the *leverage* with respect to boat A. Leverage is the distance between boats measured perpendicular to the wind direction, also shown in Figure 6.8(b). The larger the leverage, the larger the effect of wind shifts is on the positions of the boats in both positive or negative sense.

In the example, the wind shifts to the left, which is to the advantage of the boats on the left side of the fleet. If the wind had shifted to the right, the



(a) A fleet of four boats with boats A and B on the same line of equal position (LEP) and sharing the first position. Boats D and C share the second position.



(b) After a wind shift the LEPs have rotated with the wind. Boats A and C were on the left and benefitted from the shift while boats B and D were on the right and fell back. The advantage (and risk) depends on the leverage: the distance between the boats perpendicular to the wind direction.

Figure 6.8: The impact of a wind shift on the relative positions of the boats on the windward leg.

boats on the right side of the fleet would have had the advantage.

To take advantage of a wind shift, a boat has to be on the side of the fleet where the *new wind* will come from. It is the tactician's job to position the boat relative to the fleet, therefore the tactician is concerned with anticipating wind shifts. When the tactician is confident of the chosen strategy, he may choose (if possible) to increase leverage to take more advantage of anticipated wind shift as possible. However, often risk is kept small to be able to recover losses and gain little at a time. The wind graph mentioned in Section 6.1.3 is a valuable instrument for the tactician to anticipate wind shifts. Figure 6.6(b) shows how a city or hill can generate oscillating wind shifts. Understanding how to deal with that situation is part of tactics, looking for or avoiding that situation is part of strategy.

Figure 6.9 illustrates the impact of oscillating wind shifts. Both boats start at the same position with the wind from the original direction 0. Both boats are shown to have sailed the same distance measured along their course at positions A3 and B3. Boat A's tactician expects them and is looking for the shifts to use them to their advantage. Boat B's tactician is not looking for the wind shifts and is simply sailing the windward leg as they were doing in Figure 6.3.

As the wind shifts to direction 1 (positions A1 and B1), both boats experienced it as a header and boat A decides to tack. Boat B continues sailing on the same bow but their course has changed, the bow has been turned away from the old wind to their disadvantage.

At positions A2 and B2 the boats are on the same LEP (LEP I) with respect to wind direction 1. But as the wind continues to oscillate, and shifts back to direction 0, the LEP rotates with the wind and it becomes apparent that boat A has an advantage because boat B is behind on boat A's position (LEP II). Boat B tacks to stay within the laylines (of wind direction 0) while boat A continues on starboard–tack.

Boat A is aware of the oscillating winds and is expecting the wind to shift to direction 2. Ideally boat A wants to be just below the layline of wind direction 2 as it approaches the mark so it tacks in anticipation. When the wind shifts to direction 2, boat A experiences it as a lifter (their tack is lifted towards the mark) while boat B again experiences a header because they are sailing on the other bow. The header will point boat B away from the windward mark. Note that boat A was to the left of boat B while the wind was shifting to the left, they had positive leverage. To boat B's surprise in position B3, boat A is already rounding the windward mark (position A3). Boat A has used their knowledge of the presence of oscillating wind shifts to gain in the race.

A rules of thumb on the tactics level with respect to oscillating wind shift is: *Sail in the direction where the next shift is expected to come from.* This will accomplish two things. First, the boat sails the lifted tack, pointing it closer to the windward mark. Second, positive leverage is created over rivals for when the next shift arrives. In contrast to persistent wind shifts, oscillating wind shifts only last a short time and the advantage is taken immediately by sailing the tack that is lifted with respect to the average heading, as shown by boat A (A1) in Figure 6.9.

Another rule of thumb is: *Stay close to the fleet to hedge risk.* A tactician is never 100% sure of the next wind shift. By staying close to fleet a moderate advantage can be had if he's right, but not all is lost when he's wrong.

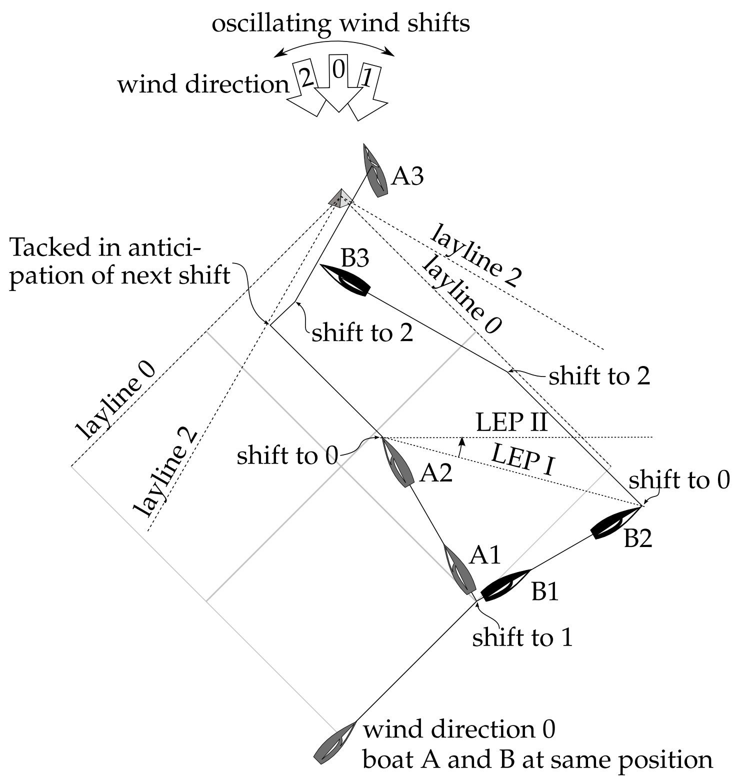
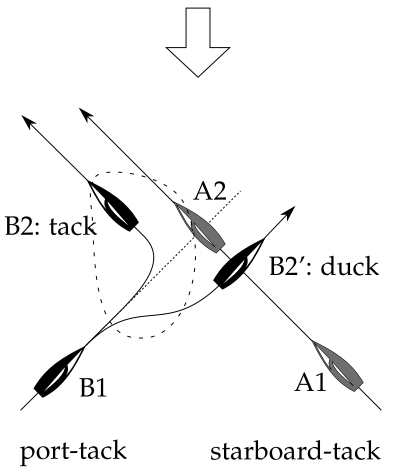


Figure 6.9: Climbing the ladder of LEPs by using oscillating wind shifts and the effect shown in Figure 6.8(b). Boat A takes advantage of the presence of oscillating wind shifts and boat B does not. Positions A3 and B3 show the advantage shifts give when effectively dealt with.

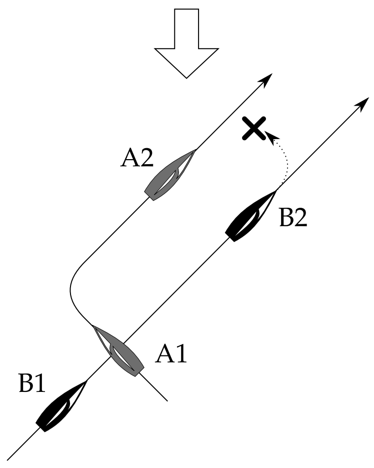
Racing rules

Tactics deals with positioning with respect to rivals. Racing rules are important at the tactics level because they allow the tactician to control other boats or to be controlled. Two of the many rules are exemplified:

> Rule 10: When boats are on opposite tacks, a port–tack boat shall keep clear of a starboard–tack boat.
>
> Rule 13: While tacking a boat shall keep clear of other boats until she is on a close–haul course (completed the tack and at full speed).



(a) Rule 10: a port-tack boat shall keep clear of a starboard-tack boat. Boat B can either tack or duck to evade boat A. The dashed line shows the disturbed air from boat B's sails.



(b) Rule 13: While tacking a boat shall keep clear of other boats until she is on a close-haul course. Boat B cannot legally tack in position B2.

Figure 6.10: Illustration of situations where respectively racing rules 10 and 13 apply.

Rule 10 is illustrated in Figure 6.10(a):boat B is the port tack boat and has to keep clear of boat A, the starboard tack boat. Boat B has two options: to tack and sail in the same direction of the starboard–tack boat(B2), or to *duck* (B2′) and continue the way they were going. A duck manoeuvre can be executed very efficiently without significant loss. A close manoeuvre tack can put boat A in the windward turbulence of boat B, requiring boat A to tack. The duck manoeuvre (B2′) is usually the recommended tactic in this situation.

Rule 13 is a rule that keeps many situations transparent. For example, a port–tack boat is not allowed to tack right in front of a second port–tack boat and then claim that rule 10 applies. However, rule 13 can also be used differently: to control other boats. Figure 6.10(b) shows a situation where boat A (A1) passes in front of boat B (B1) and then tacks to position herself windward of boat B (positions A2 and B2). As a consequence of rule 13 boat B is *pinned* in that position until boat A tacks. This is a tactical weapon that boat A's tactician can use when uncertain about their own strategy. Boat A pinned boat B, herding her along to the right side of the course. It may or may not be the favored side, it is all unclear, but boat B is unable to tack and find better conditions on the left side. Boat A is now more certain about keeping first position.

Rules are rules and as such they do not present any physical constraints on the sailors. The *protest* is an physical implementation of the rules. If boat B forces itself out of the pin and breaks rule 13 (i.e., boat A has to evade boat B's manoeuvring), boat A will protest by calling out "Protest!" and flying the protest flag. Boat B can then acknowledge the protest by completing a 720° turn, which is a penalty resulting in lost time. If boat B does not acknowledge the protest, it will be settled after the race by the racing committee and boat B will face a time penalty.

The abstraction hierarchy

Figure 6.16 shows the abstraction hierarchy for the tactics level. The levels are discussed below.

Functional purpose

The functional purpose of the tactics level is to sail the racing plan of the strategy level and use local constraints and opportunities to gain on other boats. The strategy tells the tactician where he can find the most favorable conditions. The more confident the tactician (who is also in charge of strategy) is in the strategy, the farther can be sailed to the favored side to increase the gains. A good tactician will sail to the favored side but stay close to the fleet to gain a little and to reduce the risk in case he was wrong or conditions change. In this abstraction hierarchy, the oscillating wind shifts and two racing rules to deal with rivals are exemplified.

Abstract Function

The abstract representations used for gaining position in the race are the racing square with the lines of equal position (LEPs) that form the racing ladder as shown in Figure 6.8. The oscillating wind shifts are used to climb the ladder as shown in Figure 6.9. The racing rules are found at the abstract function level. They must be applied and can also be used to *pin* other boats to keep an advantage

Generalized function

This level represents the *ingredients* of the processes at the abstract function level. Therefore, it represents the boats' positions in the race, and the oscillating wind shifts to define the rotating ladder representation. Speed and handling, represented by the speed and handling levels of control sophistication, are shown as a single concept. The ability to achieve boat speed and

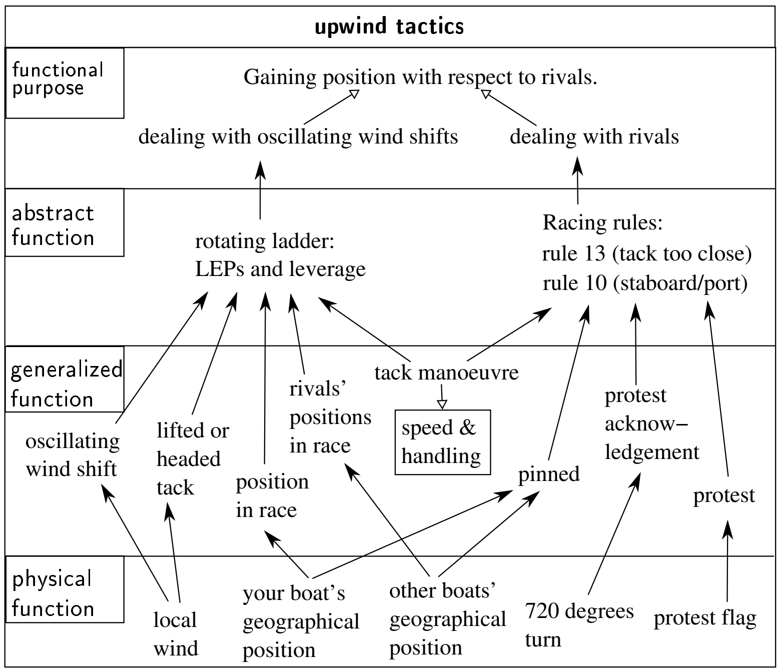


Figure 6.11: Graphical representation of the abstraction hierarchy at the tactics level of control sophistication.

handle the boat well, will determine the success at applying racing rules and the ability to apply tactics.

The concept of being pinned or to pin, is represented along with the protest and protest acknowledgement to define situations with respect to the racing rules. These are situations where the (abstract) racing rules govern behavior in a similar way that energy laws govern flight control (Chapter 3). The difference is that the laws of physics cannot be broken and the racing rules can be broken. In both cases, ignoring them as organizing principles of the work domain will lead to improper functioning.

Physical function

This level represents the above *ingredients* as the physical functions, thus how the generalized concepts are tied to their physical implementation. It includes: the wind, your boat, the rivals' boats, the 720° turn, and the protest flag. The protest and protest acknowledgement are shown as functions to implement the racing rules.

6.2.4 Boat speed level

The boat speed level deals with sailing the local conditions and tactical manoeuvres as fast as possible. To sail the windward leg, the crew has to find the speed that takes them to the windward mark as fast as possible, and execute all manoeuvring with speed in mind. The only exceptions are when safety is compromised (e.g., avoiding collisions) and when a boat finds itself in a bad position where it has to comply with racing rules before speed. Two ingredients of boat speed on the windward leg are discussed here: velocity made good and sail trim.

Velocity made good

The Velocity Made Good (VMG) is the boat speed component into the wind. As long as a boat is within the laylines, the speed that matters is the VMG. To reach the windward mark the fastest, the VMG has to be as large as possible. VMG is not perceived directly. Onboard instrumentation displays the computed VMG using wind data, boat speed and GPS data.

The combination of speed and pointing that achieves the best VMG is a characteristic of the boat. For each wind speed and angle sailed to the wind a maximum boat speed can be achieved. This is measured by sailing trials and presented in a polar diagram. An example polar diagram for a wind speed of 15 knots for a boat similar to the Baltic 45 is shown by Figure 6.12. For different wind speeds, a boat has different graphs. The graph shows the best achievable boat speed as a function of the angle that the boats course makes with the true wind direction. All speed and wind angle combinations inside the graph are achievable. Note that it is indeed impossible to achieve a boat speed straight into the wind.

The goal is to find the angle with the true wind that gives the best VMG. This is indicated in Figure 6.12 for a boat speed of 4.55 knots, sailing at a true wind angle of 51°, and achieving a VMG of 2.91 knots. The plot is used as an indication because sea conditions and wind conditions will give variations. The polar diagram shows the crew what speed and angle to aim for, and at all times the crew will work to increase VMG.

The speed polar shows the shape of the constraints: what the achievable VMG is, but it does not show how to get there. The picture is not complete. Speed is needed to generate the keel lift force that balances the sail force and keeps the drift angle small. Figure 6.13 shows the balance of forces and the

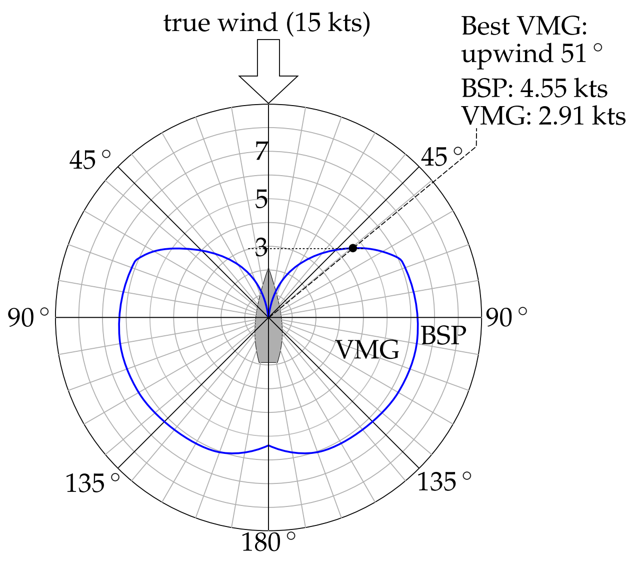


Figure 6.12: An example of the polar speed diagram shown for a wind speed of 15 knots. The best achievable boat speed is set out on the radial axis. The black dot on the graph shows the combination of the angle sailed to the true wind direction and the best achievable boat speed that lead to the best VMG for the windward leg.

drift angle. The drift angle is also the keel's angle of attack that is needed to create keel lift. A larger drift angle will cause a larger hull drag. When the helmsman is tempted to point too much into the wind before obtaining enough speed to do so, the drift angle will be large because the keel requires a larger angle of attack to generate a force that balances the sail force. As a result, the excess hull drag is preventing the boat to accelerate and build speed. The forces reach an equilibrium at a lower than maximum achievable boat speed. This is called *pinching* and can only be corrected by steering leeward (away from the wind) to accelerate the boat. After enough speed is obtained, the helmsman can steer windward and point higher, allowing the boat to settle into an equilibrium at a higher speed and lower drift angle. This is a continuous process that the helmsman is involved in when sailing close haul. To capture this process, a rule of thumb for the helmsman and trimmers on the boat speed level is: *first speed, then pointing!*

The above can be compared to the climbing performance of an aircraft and the best climb speed. A pilot who is attempting to climb at a speed below the optimal climb speed will achieve a sub–optimal climbing performance

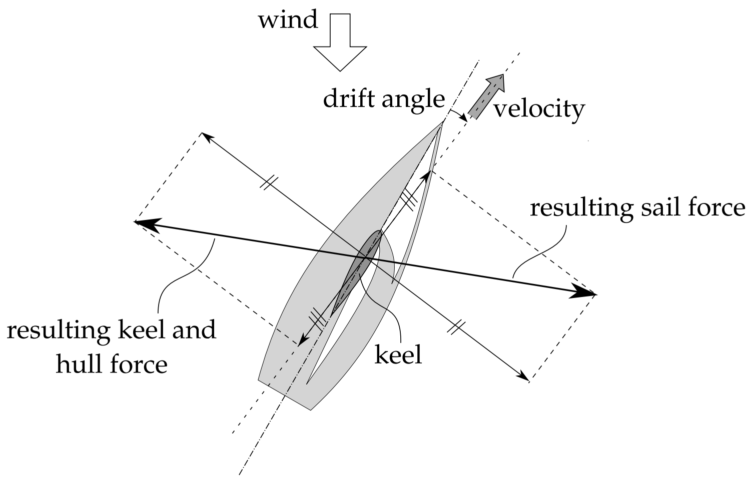


Figure 6.13: The balance of forces at a constant boat speed. The keel and hull force balance the sail force. The drift angle is the keel's angle of attack that is needed to create the keel's lift force. The drift angle also causes the hull to generate more drag. The helmsman should find the balance with the least drag by carefully building speed.

(Langewiesche, 1944). In order to increase climbing performance, the pilot should first lower the nose, decreasing climbing performance but building speed. This allows him to reach the optimal climb speed, which he can then maintain by raising the nose again. This is counterintuitive for student pilots learning to fly, who can have the tendency to point the nose high early to achieve a climb. Similar to the sailing example, equilibrium of forces can be reached at a sub–optimal point and the pilot needs to know how to reach the optimal climbing performance from a sub–optimal state. The above discussed the balance of forces and process behind reaching best VMG. How to generate the desired sail force as part of the equation is discussed below.

Sail trim

Sail trim is the process of adjusting the sail shape to generate the desired power. On every boat, each sail has is an approximate close haul sail setting that is known to the crew. It is the default setting from which the sails are trimmed by using the various controls for them. The exact sail shape is varied with the wind and sea conditions based on knowledge and experience of the crew.

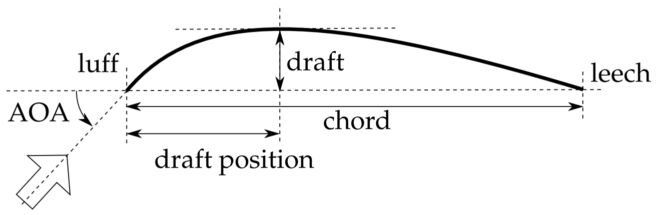


Figure 6.14: Sailshape parameters: the luff is the leading edge of the sail, and the leech is the trailing edge. The draft, draft position and angle of attack (AOA) can be adjusted with the various controls to create the desired sail shape to match the sailing conditions.

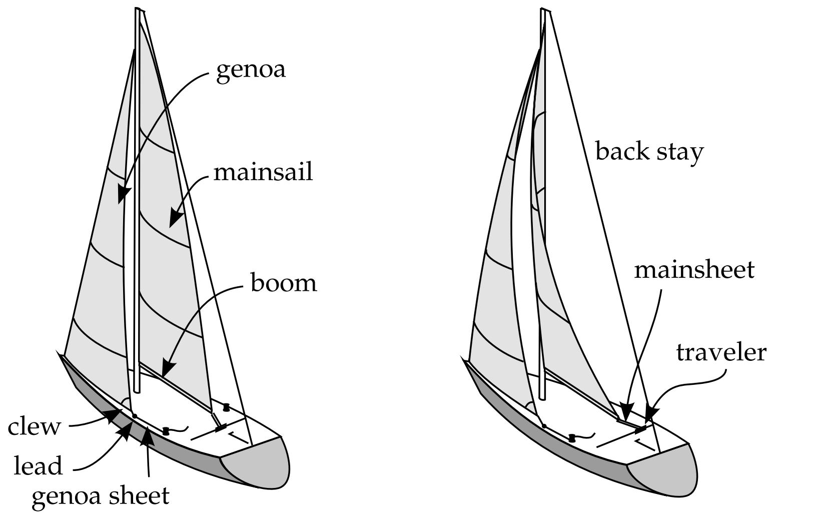
Figure 6.14 shows a cross section of a sail. The luff is the leading edge and the leech is the trailing edge of the sail. The parameters that define the cross section are: angle of attack (AOA), draft, and draft position. The AOA is mainly controlled by the helmsman through steering relative to the wind. The AOA has to be such that the airflow around the luff of the sail is non–turbulent on both sides of the sail. The helmsman is constantly watching the telltales[4] on the genoa and adjusts the boat's heading and thereby the AOA a few degrees at a time to keep the telltales flowing steadily on both sides.

The twist parameter is defined as the change in the AOA of the cross–sectional shapes of the sail from the foot up, as shown in Figure 6.15. A little twist is always needed because the apparent wind angle changes with height due to the combination of the vertical gradient of the wind speed and the boat's own speed. Twist is used by the trimmers to control the power that the sails generate to accelerate the boat.

Trimming sails can be compared to shifting gears in a car. In low gear a car can accelerate fast or climb uphill at a relative low speed. In high gear a car lacks the acceleration capability but will allow the car to travel fast. Similarly, deep sails (a lot of draft) with plenty of *twist* generate power to accelerate at the cost of pointing. Flat sails, with little twist, allow the boat to sail fast with high pointing once speed has been achieved, but lack the power to accelerate.

When sailing in chop, which is the condition of short steep waves that degrade speed, the sails are set deep, thus with draft, and with plenty of twist because the boat needs to accelerate each time it is slowed down by the

[4]Telltales are short strings attached to the sail at various locations to visualize the airflow, also see Section 6.1.3.



(a) Trimmed for high pointing: closed leech with little twist.

(b) Trimmed to accelerate: open leech with twist.

Figure 6.15: The twist parameter is important for selecting the right tradeoff between power to accelerate and high pointing and high speed.

waves. In smooth water the sails are trimmed flatter, with less twist. The best achievable VMG will be less in chop than in smooth water.

Twist is controlled with the mainsheet and traveler position. With the traveler in the center position, the main sheet pulls down on the boom and puts a lot of tension on the leech, resulting in a closed sail with a little twist, as shown in Figure 6.15(a). With the traveler to windward and the boom in the same place, less tension is put in the leech and the upper part of sail is open with twist, as shown in Figure 6.15(b). The draft and draft position of the sails is controlled by a number of controls that are not further explained: halyard tension, mast bend, sheet tension, outhaul (mainsail).

A number of principles have been explained to sail fast. The crew needs to find the best VMG to the windward mark, by building speed and trimming the sails according to the local conditions. This knowledge is mapped onto the abstraction hierarchy.

The abstraction hierarchy

Figure 6.16 shows the abstraction hierarchy at the boat speed level. The levels are discussed below.

Functional purpose
The functional purpose is to sail fast according to the tactics and strategy chosen. For the chosen subset of the domain, this includes sailing close haul and executing manoeuvres with best speed in mind. Only in exceptional cases is the crew forced to put speed at second place.

Abstract function
The boat must sail with the largest possible speed component into the wind (VMG). The boat speed polar diagram of Figure 6.12 shows what VMG is achievable for a given boat and wind speed. It represents the constraints that the boat (and equipment) imposes on sailing the windward leg, close haul. The optimal angle to the true wind is derived from it. Sails are trimmed continuously, and the helmsman drives the boat to improve VMG.

However, the balance of forces can be reached at non–optimal VMG, at a large drift angle, preventing the boat to further accelerate. Building speed is essential to arrive at the best VMG. The helmsman must know that speed must be achieved before the boat can be pointed into the wind to reach the best VMG according to the polar diagram. The *speed before pointing* rule of thumb captures the control actions needed for balancing the forces for best VMG.

Generalized function
Decoupled from their physical implementation, the discussed sail trim parameters are represented: twist, draft, draft position. These parameters hold for every boat and are, therefore, independent of implementation. The boat's speed, course, and wind are represented as the main ingredients for the abstract representations.

Boat handling, as defined at the next lower level of control sophistication, is included as a single concept. It represents that the boat handling performance propagates to this and the next levels of control sophistication. If a manoeuvre is not performed properly by crew handling, it will degrade the speed performance.

Physical function
The physically available functions to set sail shape are defined on this level. These are the physical controls available to shape the sails: main sheet, trav-

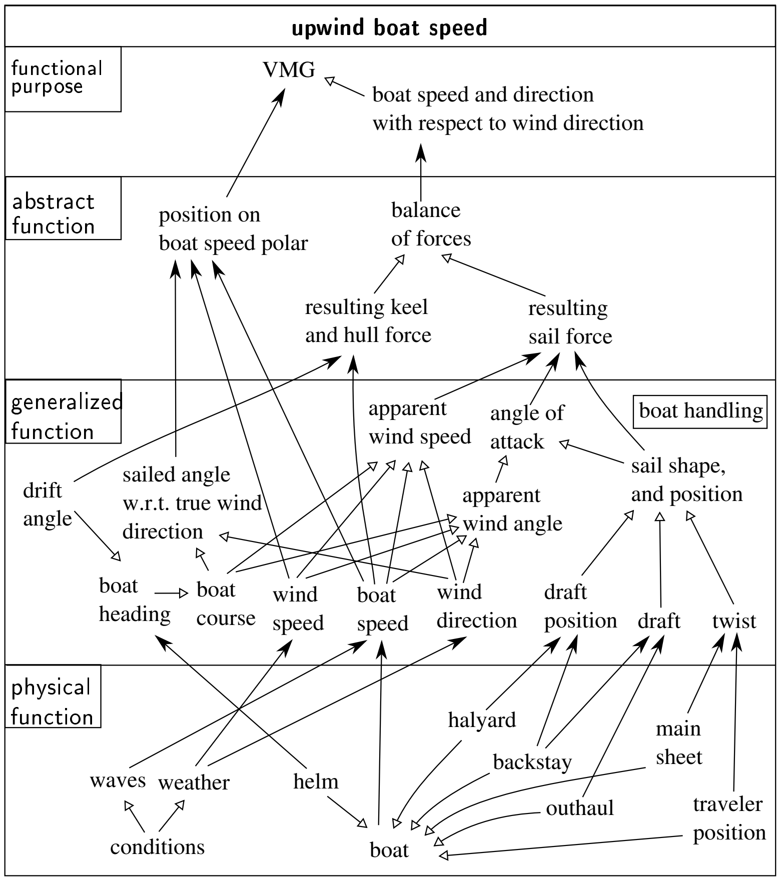


Figure 6.16: Graphical representation of the abstraction hierarchy at the speed level of control sophistication.

eler, outhaul, genoa sheet, halyards, and backstay. Each boat has these controls but how they interact exactly to reach the desired sail shape depends on the physical layout of the boat. The physical layout would be presented at the physical form level that is omitted in this analysis.

6.2.5 Boat handling level

The handling of a sail boat revolves around the defined positions. Each position comes with responsibilities, a job–description and physical work. One or more crew members are assigned to each position. If not all positions can be filled with the desired number of crew members the boat is under–

handed, which should be avoided during the short course races due to the rapid succession of manoeuvres.

Besides assigning each crew member to a position with responsibilities, all crew members have to work together as a team. One can rely on the self–organizing principles of the crew, but for high performance a number of crew organizing principles can be put in place.

With the ASA, we view the functioning of the entire system including the performance of the team to operate the boat. At this level, the crew members are the *actuators* of the work domain. Therefore, the analysis includes the constraints that (physical) performance of the crew members imposes.

According to Rasmussen et al. (1994), WDA should not include the human operator and the actions he may take. This mainly applies to the process control domain, from which WDA originates, and the role of the human operator as a supervisor and decision maker. In routine situations, the human operator is not actively involved in control.

In the sailboat racing domain, the routine actions of the crew are made part of the WDA. This can be regarded in the same way as the process routines that run inside a power plant. The routines of the crew are *designed* in a similar way and apply to the standard, routine situation. The problem solving activities of the crew are not represented in the WDA. Those would depend on the situation and would be specific to the non–routine situation.

The abstraction hierarchy

The boat handling is structured using the abstraction hierarchy. The affordances of the boat equipment, thus the opportunity they provide for achieving manoeuvring and the functions that the crew achieve are represented in Figure 6.17.

Functional purpose

The functional purpose of the boat handling level is to achieve excellent, thus quick and smooth, operation of the boat by the crew in order to sail fast. Boat handling is about the equipment, and the coordinated actions of crew members. Crew operation is optimized for sailing fast during each part of the race, to satisfy the requirement of the speed level of control sophistication.

Abstract function

Crew organizing principles are described by (Gladstone, 2006) to let the crew perform better as a team. Two of these are: *define crew positions*, and *do*

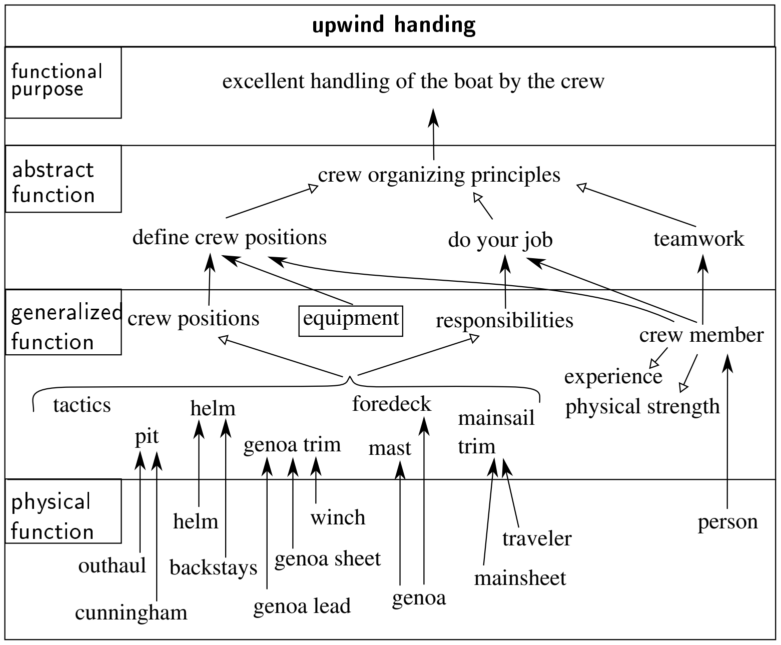


Figure 6.17: Graphical representation of the abstraction hierarchy at the handling level of control sophistication.

your job.

Defining crew positions: each crew position has a specific responsibility during each manoeuvre. For the given boat and the number of crew sailing that day, the responsibilities are defined.

Do your job: if one person is having difficulties completing a task that can create a problem. When the next person tries to help he will leave his job, not tending to his responsibilities and the problem grows. To prevent escalation, the helping hand should be well coordinated so all positions requiring tending to remain covered.

Generalized function

At the abstract function level the 'define crew positions' principle is represented. At the generalized function level this principle is instantiated by the definition of the positions and the responsibilities that go with them. Here the positions on the Beluga, sailing with a crew of 8 or 9 are exemplified. The positions are: tactics, helm, main sail trim, genoa trim (2 or 3 crew members), pit, mast, and foredeck. Each position comes with a 'job description' specifying what to do (thus also what not to do). An additional job is shared by all crew members: do your housekeeping, which means making sure that all lines run as they are supposed to.

For each job to be executed according to the plan, and achieve the function that that position provides to the racing sailboat, crew members must perform well. Two main constraints are introduced by each crew member that impacts the performance of his position. They are: experience and physical strength. Experience, is simply a property of the sailor, and more experience means quicker and more reliable execution of actions. This is mainly due to insight and coordination of actions with other crew members. Often, physical strength will degrade towards the end of the short course race because crew members become tired. This will affect the quality of handling, and consequently the speed, executing of tactics and the overall position in the race.

Equipment is also shown at the generalized function level in Figure 6.17. It represents the (proper) functioning of the boat equipment. The definition of the crew principles is based on the boat and the crew. Therefore, the 'equipment' and 'crew members' are shown to achieve 'define crew positions' at the abstract function level.

Physical function

Physical functions can be associated with each position, this is a decomposition based on the positions defined at the generalized function level rather than on components of the boat equipment.

Helm: steering of boat close haul and during tacks. When sailing close haul, the 'driver' will focus mainly on steering with respect to the wind to keep the telltales flowing smoothly along both sides of the sail.

Mainsail: trimming of mainsail using mainly the main sheet and traveler. Adjustments are made in gusty conditions, and in smoother conditions the mainsail trimmer will make small adjustments to find the best sail trim.

Genoa trimmers: trimming of the genoa using the genoa sheet and lead position. The genoa lead leads the genoa sheet from the clew to the deck as shown in Figure 6.15(a). The clew position together with the sheet tension determine twist in the genoa. The trimmers are most active after a tack to set the genoa in the close haul setting and building speed.

Pit: many lines run from the mast to the pit, in the middle of the boat, where one crew member can control them. For example, the outhaul to control mainsail draft, and the cunningham to control the mainsail luff tension. The crew member at this position is coordinating all actions with the rest of the crew.

Mast and foredeck: during the close haul course these positions require little actions. During a tack these crew members can help the genoa to the other bow, shortening the time to tack. Otherwise, action is required during sail changes or special manoeuvres not discussed here.

Tactics: keeping a close eye on rivals, wind and current. The tactician is playing the tactical game: deciding where to go and when to tack. Although not strictly a boat handling position, it is mentioned here for completeness.

Housekeeping: the deck and equipment should be in the proper and familiar state that allows the manoeuvre can be executed smoothly and quickly. If the state of the boat is undefined because routines have not been completed, the boat handling will degrade. For example, often quick tacks are needed when sailing close to rivals and responding to tactical situations. Therefore, the unused windward genoa sheet is loaded with a full set of wraps around the windward winch and the slack is taken out so it is ready for a quick tack. The used windward genoa sheet needs to be ready for a quick release, so it should be kept untangled at all times. A genoa sheet that is not released quickly during a tack will have a disastrous impact on speed after the tack.

6.2.6 Equipment

Gladstone (2000) writes that boat preparation is the hidden foundation of the performance racing pyramid. Here this foundation is made explicit with the equipment level. We list a number of points here without mapping them onto the abstraction hierarchy because when we start sailing all these must be in order. First of all, the equipment must work well to be able to race, it needs to be in a competitive condition. Sails need to be kept well and are preferably new, the underwater hull must be clean and all deck equipment such as blocks, leads, winches need to be working flawlessly.

Less obvious is that the weight distribution on the boat has an impact on speed. Below deck a lot of stuff can accumulate: sails, an anchor, crew bags, tools, etc. By concentrating all mass as close as possible to the proper center of gravity (close to the keel) and low, the moment of inertia is reduced. As a result the boat will much more easily pitch and ride the waves, and less energy is destroyed each time the bow is lifted by a wave. Of course any items that can be missed should not be on board to save weight.

On deck a lot of small adaptations can be made to enable the crew to work better and make sailing faster. One of the most useful practices pointed out by Gladstone (2000) is the marking of default positions of all main controls. For example, how tight should the genoa halyard be after hoisting the sail? A mark on the halyard will show the crew members where to aim and then make adjustments from there. Marks can be made for various locations of adjustable gear: main halyard, outhaul, genoa leads, backstay, etc. This creates a deck layout with usability in mind, for better boat handling.

6.2.7 The abstraction–sophistication analysis

The ASA is represented by a 2D matrix spanned by the levels of control sophistication vertically and the levels of abstraction horizontally as shown in Figure 6.18. Each cell contains labels of the discussed functions, concepts and constraints, giving an overview of how they work across the work domain. It shows each control problem at different levels of abstraction and emphasized that the higher levels of control sophistication rely on the achievement of goals on lower levels of control sophistication. The relation between the levels of control sophistication is illustrated by envisioning a well performing sailboat during a race and introducing a change of constraints. Three examples follow.

Crew sickness

First, we can introduce a seasick crew member who was at the mainsail trim position. His responsibility was to trim the sail and operate it during tacks, but now he is weak and unable to perform any action on board. This has an immediate effect on the boat handling level: the mainsail is not being operated. Obviously, the crew will reorganize to fill this position with another crew member. Thus, at the abstract function level of the boat handling level, the crew positions are redefined. It is decided, based on the experience of the available crew members that the tactician can serve both the tactics position and trim the mainsail. He has to pay attention to both duties and in our example we see that he does an equally good job at trimming the main sail as the now sick crew member. Therefore, this constraint, the loss of a crew member is solved at the boat handling level and does not propagate up to the boat speed level.
However, the system has changed due to a the change of position. We see

that the tactician has become less aware of what the rivals are doing. The reorganization of the crew positions, has left the tactics position weakened and this represents a change of constraints at the tactics level. Tactics are degraded due to degraded performance by the tactician, who is unable to take advantage of some of the local sailing conditions. As a result the boat finishes later than it could have done with excellent tactics.

Wind shift

Second, against all predictions in the sailing plan, the wind shifts 30 degrees without warning. Sometimes this happens. The boat is still within the lay-lines of the racing square and the crew continues sailing close haul. It is up to the tactician to decide where the next shift will come from. Will it shift further, thus should they sail the headed tack, or will the wind shift back and should they sail the lifted tack? In the first case the strategic plan changes and the tactician should revise the original plan to take the new wind into account and look for where the new advantages lie. In the second case the tactician is looking for creating leverage with respect to the rivals and plays the game of the rotating ladder. Depending on their exact position with respect to the racing square and the rivals the wind shift can be an advantage or disadvantage. This external influence impacts the tactics and strategy levels mainly, where it will be dealt with by the tactician. Resulting control actions will propagate down to the speed handling and equipment levels.

Equipment failure

Third, equipment fails. In racing sailboats this usually means a dramatic loss of time or even loss of the race. The boat is sailing on port bow and suddenly the port genoa sheet breaks. Suddenly the genoa is wildly flapping in the wind. The crew tacks to sail with the still intact starboard sheet. However, their tactical options are limited since they cannot tack until the port genoa sheet is replaced or mended. The broken equipment propagates up to the tactics level instantly. None of the tactical manoeuvres based on a tack are possible.

The first example illustrates the interdependence of human performance and system performance. Human performance is essential for performance of man–machine systems. The second example illustrates how an external constraint, not belonging to the sailing boat, activates re–evaluation of the strategic plan and tactical picture. The third example illustrates how an internal constraint, belonging to the sailing boat, at the equipment level propagates to the tactics level and constraints the possibilities there. The ASA allows keeping tracking of how constraints propagate through the system.

The nested structure of the levels allows the inclusion of functions that can only be realized through actions. Actions are not represented in the work domain analysis, so actions for boat handling are not prescribed. However, actions are needed to realize functions present on higher levels of control sophistication. For example the act of tacking is not described on the handling level but the function 'tack' is found on the tactics level. In order for the tactician to use the 'tack' function, it has to be instantiated by actions on the 'handling' level. The tactician uses the concept of 'tack' to bring the boat in another state; that of sailing on the other bow which will take it in the other direction.

6.3 Conclusions

The sailboat racing work domain is a well established work domain. The equipment and its operation have evolved into set patterns that are well described in theoretical terms and in terms of rules of thumb for quick decision making. Each race has the same clear goal: finish at the best position possible, which makes the work domain descriptions unambiguous.

In contrast to the domains described in Chapters 4, and 5, this work domain does not include automation. Additionally, the successful racing is shown to be explicitly dependent on human performance. The nested structure allows the inclusion of functions provided by human actions without describing the actions but their position and *responsibilities*. At the boat handling level the crew positions have been represented with the responsibilities that come with them. A sailor is responsible to do what is necessary to fulfil the position, and he will do that as a versatile physical *actuator* with splendid problem solving skills. Additionally, when he is unable to fulfill his position, he is responsible alert his team of the situation to prevent escalation of the situation. The responsibility (and reliability) of the humans crew allow their functions to be represented without representing situation specific actions and or procedures. The position responsibilities are invariants in the sailboat racing domain and can therefore be represented in the WDA that is not representing control solutions.

levels of control sophistication	levels of abstraction: functional purpose	abstract function	generalized function	physical function
strategy	Sailing plan: sailing in favored condition, thus on the favored side of the racing square. Risk and confidence.	Combining knowledge of local sailing conditions, and weather predictions and using them to your advantage in sailing plan.	Racing square. Wind: force, direction, oscillating or permanent wind shifts, current. $\boxed{\text{tactics}}$	Leeward and windward mark Weather Shoreline City or hills on shore Sea
tactics	Gaining position with respect to rivals. Dealing with local constraints: oscillating wind shifts and rivals.	Rotating ladder: LEPs and leverage. Racing rules: 10 and 13.	Oscillating wind shifts Lifted / headed tack Position in race, rivals position Pinned, protest, tack manoeuvre $\boxed{\text{speed}}$	Local wind Boat's position, other boats' positions. 720 degree turn Protest flag.
boat speed	Best speed to windward mark, the best Velocity Made Good (VMG).	Boat speed polar. Balance of forces: Keel and hull forces, resulting sail force.	Boat course, speed, drift angle. Wind speed, direction Angle of attack Sail shape $\boxed{\text{handling}}$	Weather Boat state: halyards, backstay, outhaul, main sheet, traveler position.
handling	Excellent boat handling: crew performance.	Crew organization principles: define crew positions, do your job, teamwork	Crew positions. Responsibilities Crew member: physical strength and experience. $\boxed{\text{equipment}}$	Boat equipment functions belonging to positions: e.g., outhaul, helm, winch,... Person(s)
equipment	Flawless operation of all equipment. Race–ready boat and equipment.	e.g., moment of inertia, weight distribution	position of sails, tools, crew bags, anchor, etc.	Marked halyards, lead positions, etc. for close haul sailing

Figure 6.18: Overview of the abstraction–sophistication analysis of the windward leg of sailboat racing domain.

This analysis also shows that the levels of control sophistication are different for different vehicle control problems. The importance of achieving boat speed for successful sailboat racing has led to this control problem being represented at its own level of control sophistication. The abstraction hierarchy at this level of control sophistication is detailed with many state variables.

Many of the rules of thumb at this level cannot be represented well in the abstraction hierarchy. For example, the influences on what the sail shape should be can be represented, but the rules of thumb for setting the controls cannot. An example rule of thumb is: when sailing in chop (short steep waves), the sails should be set deeper and with more twist to be able to accelerate at the cost of pointing. The rules of thumb are situation specific and are therefore not represented in the WDA.

Another rule of thumb that was discussed is: first speed, then pointing. The relationship between achievable speed and pointing can be shown in the polar diagram such as shown in Figure 6.12, and an explanation for this rule can be found in the balance of forces (Figure 6.13). The rule itself cannot be represented by linking 'boat speed' and 'sailed angle w.r.t. true wind angle' with aggregation and means–ends relationships. The execution of the rule of thumb belongs to the responsibility of the helm position in our example.

The only way to represent this rule of thumb in the abstraction hierarchy is to represent a model of a controller that exhibits the 'first speed, the pointing' behavior similar to the representation of the TECS control diagram at the generalized function level (see Figure 4.4). However, the means–ends relationship cannot be used as part of the model. The means–end relation is successful in linking the hypothetical control diagram to the 'first speed, then pointing' organizing principle that would be presented at the abstract function level.

In the abstraction hierarchy of the speed level, Figure 6.16, the sail shape at the generalized function level is shown to achieve 'the resulting sail force' as part of the 'balance of forces' that achieve the actual boat speed, direction (course) and VMG. Going back down to the generalized function level, the boat speed and boat course are represented as well. They achieve the position on the boat speed polar diagram at the abstract function level, which is a representation of the real VMG and the achievable VMG. Two representations mediate the achievement of VMG. The boat speed polar diagram is used in sailing to find the best VMG but it does not show how to achieve it. The additional representation of the balance of forces may provide more insight in how a better VMG could be achieved. For example, a large drift angle (see Figure 6.13) would indicate pinching and a low driving force would indicate sail trim problems.

The abstract function levels at the lower levels of control sophistication are more detailed and represented with more functional decompositions. This reflects that boat handling and boat speed are more readily represented as externalized knowledge than the tactics and the strategy levels. The latter is based on insights and understanding inside the head of the tactician and is not detailed as an externalized model. The ingredients are known (and represented at the generalized function level) but the process of combining them in a good strategy has been presented in low detail at the abstract function level.

As mentioned at the beginning of this chapter, the sailboat racing domain is a well evolved domain. All its elements are tuned to each other. The crew positions and the deck layout have evolved together. The limitation that sailing boats cannot sail straight into the wind and the boat's characteristic speed polar, as the result of design efforts, have evolved together. The need for boats to tack and the racing rules evolved together. Even the racing courses have evolved to bring the challenge to the crew to deal with all these constraints. In the end, this is what makes sailboat racing fun and exciting.

Now imagine that the introduction of some new technology, a new sail shape or hull material, would allow a sailboat to sail straight into the wind, and that the boat speed polar (Figure 6.12) loses its typical heart–shape and becomes e.g., elliptical. This technology would completely change the way in which the windward leg would be sailed. The challenges of the windward leg disappear and the knowledge that is presented in this chapter would no longer apply. A tack would no longer be performed, racing rules 10 and 13 would no longer apply, the rotating ladder loses its significance, and wind shifts no longer have the described impact on boats' positions in the race.

Technologies at the lower levels shape the higher levels of control sophistication. The ASA presents this knowledge in a structured way but it does not represent how the domain has evolved. A lot of details have been presented and most of those cannot be captured in a graphical abstraction hierarchy and must be captured by the accompanying text. This chapter presented an inventory of *representations* needed to understand how to control a sailboat in a race.

7 Discussion and conclusions

In the previous four chapters, the application of Work Domain Analysis (WDA) has been studied in four different case studies. None of the analyses are complete examples of WDA but each contributed insights into the main research questions introduced in Chapter 1. This chapter combines the generated insights from the case studies. Their combination allows us to look at the bigger picture and to come to a more generalized view on the development of WDA for Ecological Automation Design (EAD).

First, the case studies are recapitulated to show how they form the red line of the presented development of WDA for EAD. Then, a discussion reviews and compares the analyses on a number of critical points. Finally, the conclusions are presented and recommendations for future work are made.

7.1 Recapitulation of the case studies

Chapter 3 – Energy Augmented Tunnel In the Sky display

The manual control task for longitudinal flight was analyzed using the abstraction hierarchy. The analysis showed how the energy based control coupling is the abstract principle that mediates the control inputs and the speed and path goals of the pilot. In this case, the work domain was delimited by a tight system boundary, only a part of the total aviation domain was taken in the analysis. A tight system boundary around the task allowed the analysis to focus on the energy management principles of longitudinal flight. A broader system boundary, as was first attempted (Flach et al., 2003), defies the analysis and does not allow the analyst to uncover the importance of the energy management problem. Based on the analysis and by applying Ecological Interface Design (EID), the energy control actions and energy control goals are graphically represented a novel display based on the tunnel–in–the–sky display. A preliminary study indicated that the additional energy management information shown in relation to the control actions and control goals helped pilots to fly the approaches.

Chapter 4 – Analysis of the Total Energy Control System

In this case, an existing control system design, the Total Energy Control System (TECS), was analyzed. The analysis of TECS dealt with the same longitudinal flight control task that was the topic of the analysis in Chapter 3, but this time for automated control.

The available knowledge on TECS was mapped onto the abstraction hierarchy. To take the automated functions of TECS into account, a larger work domain boundary was chosen than in Chapter 3. This work domain contains more control problems and more constraints that are introduced by the automated part of the system. Other design principles needed to be satisfied in addition to the energy management principles. The automated functions of TECS: energy based decoupling, the mode hierarchy and overhead control, added more constraints and structure to the work domain than the aircraft alone. The energy based control decoupling of TECS was shown to be designed to fit the energy coupling of speed and altitude that is inherent to the aircraft. The mode logic comes from functional requirements; the aircraft has to be able to comply with ATC. The nested structure of the control problems was identified and the first step towards representing them as nested work domains with the Abstraction–Sophistication Analysis (ASA) was made.

Chapter 5 – Design of a mini–UAV system

The insights that were gained from the previous studies were applied to the WDA during to the design and development of a mini–UAV system from scratch. This work domain is characterized by the nested structure of control problems and the high level of automation that is required.

The analysis started with a *first chunking* to take the project constraints into account. Then, the focus was put on the mission capabilities of the mini–UAV system and the levels of control sophistication were identified to capture the nested structure of control problems. The levels of control sophistication allow the individual control problems to be analyzed separately. The abstraction hierarchy was used for the WDA at each level of control sophistication. Combined, the resulting structure is the two–dimensional matrix of the ASA.

The analysis had a strong focus on the automated solutions per level. It was shown that automated functions can be included in ASA. The ASA allowed the incremental inclusion of automated functions per level as they were designed, while keeping track of how the newly introduced constraints by the automation changed the work domain. The work domain description plus the automation introduced constraints, together, form the work domain description for the next higher level of control sophistication.

The ASA was used to track constraints through the represented system and visualize dependencies in the system. The propagation of constraints could be tracked both across the levels of abstraction and across the levels of control sophistication.

Chapter 6 – The sailboat racing domain

The fourth and final study took WDA out of the aviation domain and into a new vehicle control domain, that of sailboat racing. This domain has the nested structure similar to the UAV control domain but lacks automation. The WDA of Chapter 5 had a strong focus on representing the designed automation and neglected the representation of the work domain itself. The analysis of the sailboat racing domain gives insight in the nested structure independent of automated control. For each control problem, abstract representations were found that help the human crew to control their boat during the race. The ASA could be applied to successfully represented work domain knowledge in this non–aviation and non–automated vehicle control domain. Furthermore, the invariant properties of human performance were made a part of the ASA by including *boat handling* as a level of control sophistication, which would not be possible with the original WDA. Human performance is a key to sailboat racing and its representation in the analysis is essential for a complete representation of the domain.

7.2 Critical discussion

The previous chapters, four different analyses were represented. Each chapter presented an in–depth analysis of the work domain and the control problems to be solved. Each study contributed with insights into the application of WDA for automation design. The analyses were performed in an iterative fashion where insights that were obtained from one domain could be applied to another domain. We look back at the analyses and critically discuss selected points with the aim to come to a generalized approach.

7.2.1 The structure of the work domains

The domains that were subject of the presented studies have a nested structure of control problems in common. In Chapter 3, the system boundary was chosen tightly around the manual longitudinal flight control task to exclude other control problems concerning flight. In the other chapters, the scope of the analysis could not be limited in the same way, and we were presented with multiple control problems per domain.

A *first chunking* can be performed to identify the main functions that govern the work domain before dividing the complete work domain over levels of control sophistication. In Chapter 3, a first chunking was not presented but the first attempts to WDA with a larger system boundary (Flach et al., 2003) led us to choose a tighter work domain boundary. A first chunking was applied in Chapter 4 to produce an overview of the available knowledge on TECS. Then, it was applied in Chapter 5 revealing that project constraints are shown to be part of the design space as well. For the UAV system of Chapter 5, the first chunking allowed us to visualize that the choice of some components was based on their commercial availability and not their technical specification. In Chapter 6, the first chunking for the sailboat racing domain was skipped because the nested structure could be adapted from Gladstone's (2000, 2006) racing pyramid. A first chunking is essentially the mapping of available knowledge on an abstraction hierarchy before starting the in–depth WDA. The first chunking is then used to identify and separate the nested control problems in the work domain.

Levels of Control Sophistication

The nested structure of the work domains is captured in levels of control sophistication. The possibility of having nested abstraction hierarchies was already described by Rasmussen (1998) and the need for representing nested dependencies was also addressed by van Paassen and Mulder (2004). The nested structure found in the vehicle control domains was made explicit by introducing the levels of control sophistication in WDA in the presented case studies. The term 'levels of control sophistication' is chosen to reflect that the outer levels of automation in the nested structure provide more sophisticated control than the inner levels. The choice of levels of control sophistication is based on the identification and separation of control problems.

For TECS, in Chapter 4, the choice of the levels was based primarily on the structure of the control diagram of TECS, resembling the inner and outer loop control functions. However, the separation of the aircraft independent part of the TECS core and the aircraft tailored part, does not seem to benefit the analysis of the energy management principle. These two parts have a strong coupling through the preservation of the energy relationship between the signals and the bandwidth separation principle (see Figure 4.4). If the control problems have a high degree of coupling, e.g., through gain dependencies, they should not be separated.

To further illustrate, at the *boat speed* level of the sailboat racing analysis in Chapter 6, the achievement of VMG and sail trim are represented. Both are needed for speed but one could reason that proper sail trim is needed for sailing a good VMG and see a nested dependency here. In reality, sail trim and steering are closely coupled and splitting the *boat speed* level in two levels would not have benefitted the analysis.

For the UAV domain we found that, higher levels of control sophistication allow the inclusion of functions that are provided by automation on the lower levels of control sophistication. For the studied domains, the levels of control sophistication differ from each other by the different representations that are needed to solve the particular control problems. In hindsight, with the lessons learned from the other case studies, the choice for levels of control sophistication should be based on the decoupling of the control problems. Remembering that each level of control sophistication is a nested work domain with a boundary, it is advised to choose the boundaries such that there are fewer interaction across them, similar to what was defined for choosing the system boundary (discussed in Chapter 2).

Lower levels of control sophistication have a tight coupling to the physical world, while higher levels are based on theoretical constructs to help achieve sophisticated tasks. We see this in TECS, the lowest level represents the aircraft and the higher levels represent the mode hierarchy and mode coordination logic. In the UAV domain we see that flight is represented at the lowest level, while the higher levels deal with navigation and mission achievement. The coupling to the physical world is less on the higher levels where navigation and mission are represented. In the sailboat racing domain we see the same pattern: the equipment is at the lowest level, then comes the handing of the equipment, then the physics of sailing fast, then the tactics that concerns non–physical sailing rules, and finally strategy that combines all these representations into a sailing plan. The higher levels have in common that they introduce representations that are not bound as much by physical laws as by non–physical organizing principles. These depend more on how the intentions in a work domain are organized than on the imposed physical organization of the system.

7.2.2 The abstraction hierarchy analysis

The levels of abstraction have been adopted from Rasmussen's definition in the process control domain. Throughout this thesis a substantial argument to deviate from this definition has not been found. The analyses were limited to the upper four levels mainly because the physical form was of little interest, it was often unknown, or the description would be tedious and superfluous with detail with only little contribution to the development of WDA.

Part–whole decomposition and aggregation

"Normally, the domain representation is intended to be independent of task and situation and, in such a case, a decomposition must be considered separately for each level of abstraction." (Rasmussen et al., 1994) (p.44). Unfortunately the Abstraction Decomposition Space (ADS) is most often used as a two–dimensional matrix where the levels of decomposition cut across the levels of abstraction (Burns & Hajdukiewicz, 2004; Vicente, 1999). This often produces a diagonal across the two–dimensional ADS, effectively reducing the two–dimensional space to a one–dimensional line.

In this thesis, this particular application of levels of part–whole decomposition has been found impractical and decomposition is applied per level of abstraction. The part–whole decomposition depends on the concepts, functions and constraints that are identified at a level of abstraction. At the lower levels of abstraction, the part–whole decomposition applies to physical components and at the higher levels of abstraction, the part–whole decomposition applies to non–physical functions. Additionally, the decomposition relationship is presented through *aggregation*. The direction of aggregation is compatible with the direction of abstraction since both relationships lose detail and gain meaning with respect to the system's purpose.

The levels of abstraction

"What goes on which level precisely?" was a challenging question in the process of analysis of all four case studies. Examples from the original CSE domain, process control, gave some support but left the question mainly unanswered for the domains of this thesis. Process control examples focus on the physics underlying control, which results in the representation of mass and energy balances at the abstract function level. This is indeed useful for the lower levels of control sophistication, but not for the higher levels of control sophistication. By reviewing the analyses of the cases studies, the following descriptions are given for the functional purpose level and the abstract function level complementary to the definition given in Chapter 2.

The **Functional purpose** level represents the purpose of the level of control sophistication of the system being studied and all underlying levels of control sophistication. At this level the requirements on the control are formulated. Often a single function was represented that indicates the main purpose of the represented part of the system. As shown for the UAV domain and TECS, this purpose can be decomposed into 'product', 'safety', and 'efficiency' (van Paassen, Mulder, van Dam, & Amelink, 2005). This was not done for the analysis underlying EATIS and the sailboat racing domain, where the analyses focussed on the purpose associated with product only. The functional purpose level of the first chunking of TECS, in Figure 4.4, represents the requirements associated with the design goals of TECS.

The **Abstract function** level of power plant examples show 'mass and energy balances' (Rasmussen et al., 1994; Vicente, 1999). In Chapter 3, the energy management problem also nicely fitted this description. However, we found that abstract functions are not limited to physical laws by definition. Physical laws are found in the lower levels of control sophistication but at the higher levels we found artificial constructs to deal with e.g., 'tactics', 'mission', and 'control authority'. These are not organized around physics but around non–physical constructs that are invented to structure the more sophisticated control problems.

The most powerful description that Rasmussen et al. (1994) give of what belongs at the abstract function level is: *organizing principles*. These are the principles around which the solution to a control problem is organized. It gives the analyst the freedom to represent physical laws, legal laws, algorithms, design provisions and other principles.

The introduction of automation in the system made us think about the representations that allowed computers to solve the problems (e.g., the A* algorithm and cost map in the UAV domain in Chapter 5). The representation of algorithms and input–output transformation is inspired by (Marr, 1982), who recognized that every computational unit has to be understood by what it tries to achieve and how the computation is physically implemented. Marr's middle level is the algorithmic level and fits well with the abstract function level representing automated control.

In retrospect, the abstraction hierarchy that is presented in Chapter 3 for the manual longitudinal flight control task differs from this definition. By using the insights gained from the other case studies, the content of the levels should be shifted to have all energy representation at the abstract function level. The cause–effect diagram would not be presented at the physical function level but at the generalized function level. Figure 7.1 shows the graphical representation of the abstraction hierarchy with these changes. Shifting the content of the levels does not change the underlying analysis.

The means–ends relationship requires interpretation

The way in which generalized functions achieve abstract functions is not specified by the means–ends relationship. For example, the 'amplitude limit' in the speed signal path in the mode hierarchy of TECS achieves 'control authority' at the abstract function level as shown in Figure 4.4. This bit of knowledge explains the reason for this 'amplitude limit' in the design. However, it does not explain how the 'amplitude limit' achieves 'control authority'. To understand this relationship, the analyst needs to understand the role of the 'amplitude limit' in the control diagram, and the concept of 'control authority' in relation to the purpose of TECS. When both are under-

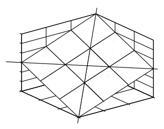
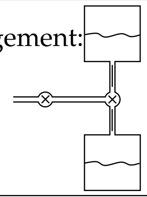
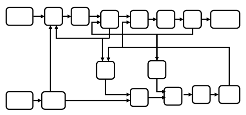
LEVELS OF ABSTRACTION	
Functional purpose	fly trajectory: follow speed profile; follow altitude profile
Abstract function	energy awareness:  law of conservation of energy — energy management: 
Generalized function	controlling the state variables — cause-effect of control for a generic aircraft 
Physical function	aircraft components: wings, fuselage, control surfaces, ...
Physical form	component shapes, materials, ...

Figure 7.1: The 'new' abstraction hierarchy for the manual longitudinal flight control task of Chapter 3. The content of the levels is brought in line with the insights gained from the other case studies presented in this thesis.

stood, the representations at different levels of abstraction are connected in the mind of the analyst and an understanding is formed.

Reasoning with the abstraction hierarchy requires the human mind to make these connections. Therefore, the intelligence (that allows reasoning) is in between the levels of abstraction. The means–ends relationships in the abstraction hierarchy shows where to look for the connections. As such, the abstraction hierarchy is a knowledge map, and like a street map it requires interpretation to work with it. The means–ends relationships are powerful in guiding the human interpreter to make links between different representations and come to an understanding that allows reasoning. Therefore, intelligence is not a property of the means–ends network nor of the abstraction hierarchy. The generalized and heterogeneous nature of the knowledge represented in the abstraction hierarchy requires a more versatile interpretation than a street map.

Recursive dependencies

In Chapter 6, at the boat speed level of control sophistication the same functions: 'boat speed' and 'boat direction' are represented at multiple levels of abstraction: the functional purpose, abstract function, and generalized function levels (Figure 6.16). There was a natural tendency to want to represent the recursive dependency between speed and direction to achieve the optimal VMG. However, this dependency could not be effectively represented by the abstraction hierarchy. At each level of abstraction, the same functions mean something slightly different. The generalized function level represent the boat speed and boat direction in terms of vehicle dynamics. At the abstract function level, the boat speed polar is used for comparing the actual boat speed and boat direction (the velocity made good, VMG) and the achievable speed and direction (Figure 6.12). The balance of forces can be used to understand how to improve the speed and pointing (Figure 6.13). At the functional purpose level, the purpose of the boat speed level is again expressed in boat speed and boat direction, since best VMG is the purpose during the windward leg of the race. Thus, the levels of abstraction hold different representations for understanding the control problem but does not show how to control the boat. This is the nature of the abstraction hierarchy. Indeed, control over speed and direction is required to sail with a better speed and direction: better VMG at the functional purpose level. The art of doing this well is captured by the *speed before pointing* rule of thumb but it is not captured by the abstraction hierarchy. The abstraction hierarchy represents the reasons for design, where the means–ends network across the levels represents the achievement of the purpose of the system independent of run–time state, situation or particular actions. It is a knowledge representation independent from goals and actions, and as explained by (Vicente & Rasmussen, 1992), the abstraction hierarchy is the representation at the knowledge based reasoning level of the SRK taxonomy. The rules of thumb belong at the rule based level of the SRK taxonomy and are therefore not representable in the abstraction hierarchy.

Where to start with the abstraction hierarchy?

Rasmussen et al. (1994) suggest to start with the middle level, the generalized function level that contains the most familiar terms for a particular domain, and work outward to the least and most abstract levels. (Vicente, 1999) suggests to start with the functional purpose level and the physical function level, and then to fill in the intermediate levels.

Our experience is that the approach of (Rasmussen et al., 1994) works best for the higher levels of control sophistication, and that Vicente's (1999) approach works best for the lower levels of control sophistication. The lower levels have a stronger coupling to physical objects. The inventory on the physical objects and influences is a good starting point. Higher levels have a weaker coupling to the physical world and are based on non–physical constructs (e.g., mode hierarchy, mission, navigation, tactics). Their representation, which is independent of physical implementation, can best be started at the generalized function level. Their coupling to the physical world can then be represented at the lower levels.

7.2.3 The abstraction–sophistication analysis

Three kinds of relationship are used in the ASA. First is the nested relationship of the levels of control sophistication. Second is the means–ends relationship across levels of abstraction within a level of control sophistication. Third is the aggregation relationship that supports functional part–whole decomposition within a level of abstraction. These should not be confused, below the relationships are compared.

Abstraction and control sophistication

Care should be taken to not confuse the levels of abstraction and levels of control sophistication. Along the dimension of control sophistication of the ASA, abstraction of the represented control problems takes place. For example, in the UAV domain, controlling a *mission* is a more abstract way to control the system than controlling *navigation* of UAVs. Similarly, *flight control* is more concrete than *navigation*. Control sophistication is a form of abstraction. The levels of control sophistication deal with abstraction of the complete control problem, while the levels of the abstraction hierarchy deal with abstraction of the functions that belong to each control problem.

By reviewing the definition of the abstraction hierarchy given in Section 2.3.1 we see that the levels of control sophistication have clear similarities to the abstraction hierarchy. For example, control over the UAV system at the mission level involves more abstract goals than control al the flight level. The nested structure of the levels can be interpreted as the means–ends relationship, the why–what–how relationship can be used across the levels.

"*How* does a UAV navigate to point A?" – "by computing a path in this direction." "*Why* does a UAV navigate to point A?" – "to achieve a mission element."

However, there is a difference. The abstraction hierarchy represents the entire work domain at each level of abstraction, which is the first property of a stratified hierarchy that was discussed in Chapter 2. In contrast, the levels of control sophistication were introduced to represent *parts* of the work domain and to incrementally analyze the work domain. The levels of control sophistication represent nested work domains while the abstraction hierarchy represents a single delimited work domain.

Levels of control sophistication and levels of decomposition

The levels of control sophistication and levels of part–whole decomposition also have some similarities. In some examples the choice of levels of part–whole decomposition in the ADS resembles the levels of control sophistication. We discuss this with Rasmussen's (1998) analysis of a UAV system's SEAD (suppression of enemies air defense) mission. Rasmussen starts with the ADS, where the levels of part–whole decomposition cut across all levels of abstraction. The levels of part–whole decomposition are defined as: national level, theater of engagement, active force/air force, SEAD mission, and UAV team. These levels capture the nested organizational structure around the execution of the mission, like the levels of control sophistication capture the nested structure of control problems.

The initial ADS is detailed with *organizational units and actors* along the diagonal of the ADS. This diagonal disappears when each of the levels of part–whole decomposition is presented with a complete abstraction hierarchy. However, linking the abstraction hierarchies in the ADS is not further specified other than the non–linear relationship as was discussed in Chapter 2. Furthermore, Rasmussen (1998) illustrates that each of the organizational units that were mapped onto the ADS can be analyzed with a nested abstraction hierarchy, indicating that multiple dimensions of abstraction are needed to present the complete work domain.

In hindsight, the ASA for the UAV system in Chapter 5 and Rasmussen's (1998) analysis map the same nested structure of the work domain. In these approaches Rasmussen has reshaped the ADS to accommodate the nested structure of an organization, while the work of this thesis has added a new dimension to the abstraction hierarchy to capture the nested structure. The

ASA has a more systematic approach to defining the relationship between the abstraction hierarchies at different levels of control sophistication.

Analysis for system design benefits from keeping the dimensions of control sophistication and part–whole decomposition separated. Decomposition is applied per level of abstraction, while abstraction is applied per level of control sophistication. Additionally, part–whole decomposition is component oriented and does not capture the structure of nested control problems.

The space spanned by the ASA

The space spanned by the ASA can be characterized by four areas as shown in Figure 7.2. At the bottom right we find the inventory of physical and generalized functions that belong to the system. In the presented domains, these were for example the UAV system and the sailboat equipment. In the upper right corner we find an inventory of elements that are external to the system. Examples of these are a target that is observed by the UAV (mission level) and the markers that span the racing square in sailboat racing (tactics and strategy levels). At the bottom left, the purposes and governing principles are of a physical nature, relating to the physics of the system. Moving up to the upper left corner, we find purposes and governing principles that are of a non–physical nature. In the presented domains, these were for example, the ATC speed rule in TECS, the missions elements in the UAV domain, and the racing rules for sailboat racing.

In–between these areas we find a mixture of these characterizations. For example, the modes of the TECS mode hierarchy represent a mix between the physical possibilities of flight path control and ways to control the path that make sense to the pilot, e.g., 'altitude hold' and 'glide slope' modes. In the UAV domain, the navigate level is a mix of physical positioning and the invented method of defining waypoints to define a flight plan. The number of levels of abstraction and the number of levels of control sophistication are not fixed, and depend on the interest of the analysis.

Science or fiction?

Based on the experience with WDA, and without providing proof or an analysis, it is claimed that it would be possible to represent the science–fiction knowledge of a well structured science fiction story in WDA. Within the science–fiction story the development of events is in fact constrained by a

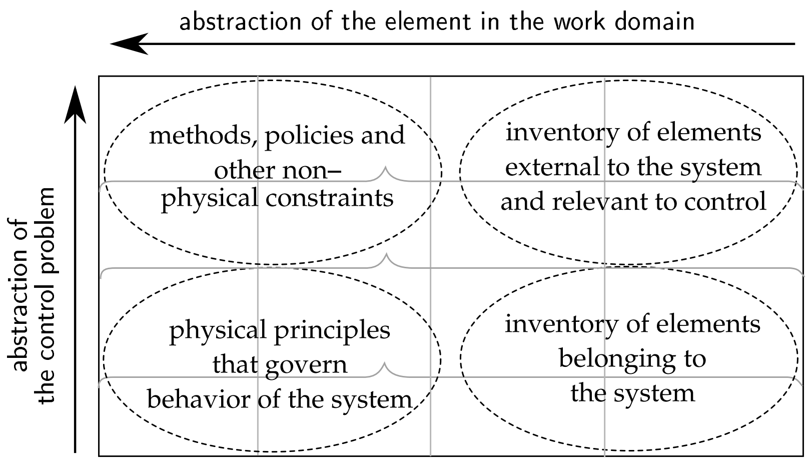

abstraction of the element in the work domain
abstraction of the control problem
methods, policies and other non–physical constraints
inventory of elements external to the system and relevant to control
physical principles that govern behavior of the system
inventory of elements belonging to the system

Figure 7.2: The abstraction–sophistication analysis spans the problem space with two dimensions that are both a form of abstraction. The vertical dimension deals with abstraction of the control problem, while the horizontal dimension deals with abstraction of the components that belong to a control problem. The areas in the spanned space can be characterized by the shown descriptions.

fictive world with fictive technology based on fictive science. Knowledge of how the fictitious world is structured is often the key to understanding the plot. This knowledge can well be presented at the various levels of the ASA. However, this does not provide the basis for verifying truth of the represented knowledge. Thus, WDA provides a way to structure knowledge but does not provide a means to verify the truth.

Experiences with WDA

The following list provides an overview of our experience with WDA in the previous chapters:

- WDA was not a cookbook recipe for eliminating complexity. It was a framework that allowed the modeling of a work domain. It required a deep understanding of the work domain and intelligent modeling.
- WDA was an analysis tool to explore those structures and functions that were relevant to the analyst/designer at that time.

- The levels of control sophistication expanded the analysis space to include a larger work domain with multiple, nested control problems. This was illustrated well by the example in Appendix A.
- More sophisticated control implied more constraints in the analysis. Each level of control sophistication held representations of a more sophisticated control problem than the lower ones, introducing new constraints.
- Through the layered structure of the ASA, automation could be included incrementally in WDA.
- It was inevitable that automation introduced constraints in the work domain. Computers and algorithms required different representations for control than human beings.
- The ASA represented knowledge, and like a street map it needed intelligent interpretation to be useful.
- The ASA did not provide a systematic check for proof of the represented knowledge. This needed to be provided by the analyst and designer.
- The ASA demonstrated that multiple dimensions of abstraction were relevant in the vehicle control domain: abstraction of the control problem and abstraction of the elements that belong to the control problem. Other domains may require different or more abstraction dimensions to span the problem space well.
- The graphical abstraction hierarchy offered a very limited space to include all domain knowledge. It was useful as an illustration of how a selection of functions were connected but it could not represent the whole analysis effectively. The graphical abstraction hierarchy served as an illustration of the accompanying text–based analysis.
- The generic nature of the abstraction hierarchy allowed its representation to be bent to accommodate what the analyst wishes to represent. This is, however, undesirable.
- The abstraction hierarchy was not intended for the modeling of recursive dependencies, closed loop control, or the dynamics of a system.
- As part of EID, the means–ends relationship linked functions to their purpose in the system, supporting the operator. As part of EAD, the means–end relationship linked functions to their reasons for design, supporting the designer.
- The abstraction hierarchy and the ASA were used to make an initial

inventory of the work domain. They allowed the main tradeoffs and reasons for design to be represented. However, WDA should not become more important than the design effort itself.

7.3 Conclusions

This thesis has presented four case studies into the application of WDA and its adaptation to the design of automation. The successes and limitations of the approach are presented.

7.3.1 Successes

The clarity of an abstraction hierarchy analysis benefits from a limited scope of the work domain that is analyzed. The ASA allowed a work domain to be divided into nested work domains, each addressing a separate control problem through the application of the abstraction hierarchy. Each abstraction hierarchy had a boundary that was chosen tightly around the control problem. The limited scope allowed us to address the specific constraints of each control problem with little clutter from other constraints. Combined, the nested abstraction hierarchies provided a much larger problem space to be represented than a single abstraction hierarchy.

Lower levels of control sophistication represented physical constraints, such as the conservation of mass and energy. Higher levels of control sophistication mainly represented non–physical constraints, such as policies, rules, and operating procedures. The levels of control sophistication were not a feature of the designed automation but are rather a feature of the work domain itself. However, the evolution of the work domain (and the way of working) is affected by the evolution of the technology and the automation in it.

Each level of control sophistication was seen as a work domain with a limited scope within the complete work domain. This structure allowed the incremental inclusion of automated functions in the analysis, which in turn allowed the analyst and designer to keep track of how the work domain was transformed by the automated functions. The ASA also allowed the inclusion of functions that were based on human performance, as shown in Chapter 6. The ability to include these in WDA is valuable because the limitations of the automation and human actors can then be taken into account at higher levels of control sophistication.

Visualizing constraint propagation helped to keep track of couplings, thus complexity in the work domain. Changes in the system properties propagated up through the levels of abstraction and the levels of control sophistication. Changes in requirements for control propagated down through the levels of control sophistication and abstraction.

WDA was not a phase in the design process but an integrating force that constantly kept track of how the designed control solutions match the representations of the control problems, which was also found by Militello, Dominguez, Lintern, and Klein (2009). WDA is seen as the scaffolding that is needed to build a high–rise building. Both grow at the same time and both support each other during growth.

7.3.2 Limitations and recommendations

There were limitations with the abstraction hierarchy and consequently with the ASA. Performing WDA required a significant amount of time figuring out which functions belong to which level exactly, as was pointed out by Vicente (2002). According to Lind (2003), this is mainly due to the lack of guidelines for modeling with the abstraction hierarchy. The beneficial result was that the analyst became very familiar with the work domain.

The means–ends relationship was not effective at modeling all system properties. A main limitation for representing (automated) control was that recursive dependencies found in closed loop control could not be represented across levels of abstraction. However, the control loop could be represented at a single level of the abstraction hierarchy and the characteristics of its functioning (what it achieves) could be represented at a higher level of abstraction through a means–ends relationship.

Limiting complexity was not found to be an inherent feature of the abstraction hierarchy or the ASA. The designer/analyst used the evolving analysis of the work domain in combination with design expertise to make design choices that limit the automation introduced complexity.

The theoretical case is built that WDA can be used to limit automation induced complexity. An evaluation of how the approach limits complexity has not been found feasible because the creative design phase after the analysis significantly contributes to the success of the system. To further illustrate we can view an alternative to the EATIS display (of Chapter 3) presented by Catton, Starr, Noyes, Fisher, and Price (2007). Their display is based on a different analysis of the energy constraints and has a different visual design. A experimental evaluation comparing the displays will most likely give different results, but are these due to the analysis or due to the creative design? An approach is needed to be able to quantify how WDA impacts the performance of the pilot using the display. Similarly, for automation design an approach is needed to be able to quantify the impact of WDA on the man–machine system's performance.

A pitfall for WDA for automation design is to focus on the representation of the new automation, and not the representations of the work domain. We experienced this pitfall first hand during the design of the UAV system. As proposed in Chapter 5 and by Lind (2003), the analysis should be split into a work domain–only part and an automation part. Per level of control sophistication this involves the representation of the work domain based on the lower levels of control sophistication, the representations needed for control, and the representations of the automation introduced constraints. Figure 5.24 showed this approach for the UAV domain, and Figure 7.3 shows a generalized view of this approach. The emphasis was put on the work domain constraints independent of automation design to make them leading in the ecological approach (as argued in Chapter 1). The analysis of the UAV domain in Chapter 5 showed mainly the mapping of automation introduced constraints while the analysis of the sailboat racing domain in Chapter 6 showed the mapping of work domain inherent and automation independent constraints. The ASA supported the representation of both steps, although this has not been explicitly exemplified in a single work domain analysis in this thesis.

7.3.3 Final remarks

The ASA offers a different view on system design and the design space. It helped us to identify and analyze the representations that are needed for solving the control problems in a work domain. The addition of the levels of control sophistication shifted the focus away from the physical governing principles, and explicitly included the non–physical governing principles that dominate the more sophisticated control. Furthermore, the nested structure found in the vehicle control domain is made explicit in the analysis. The extension of work domain analysis with the levels of control sophistication has lead to a richer representation of the work domain than a single abstraction hierarchy or the abstraction–decomposition space.

Whether it is for interface design (as in EID) or automation design (as in

$$\overbrace{\underbrace{\left.\begin{array}{c}\text{resulting work domain of this level}\\ = \\ \text{automation introduced constraints}\\ + \\ \text{control problem independent of automation}\\ + \end{array}\right\}}_{\text{lower levels of control sophistication}}}^{\text{higher levels of control sophistication}} \begin{array}{c}\text{the level of control}\\ \text{sophistication under}\\ \text{consideration for}\\ \text{automation design}\end{array}$$

Figure 7.3: Including automation in WDA is a stepped process. The lower levels of control sophistication are the work domain of the design effort. During the design, first representations are added that capture the control problems of the levels of control sophistication under consideration. Second, the representations that are introduced by the automated control solutions are added. Together they result in the work domain of higher levels of control sophistication.

EAD), representation is a powerful means to reduce *apparent* complexity. A thorough analysis and representation of control problems in a work domain requires more expertise, time, and funds, which has been addressed as a drawback by Vicente (2002). On the other hand, with the trend of increasing automation, methods are needed to manage complexity to improve safety and to make new human–machine system capabilities possible. The application of the ASA enabled us to analyze and visualize the elements of the work domain and how they contribute to complexity of the system prior to and during its design.

Bibliography

A123 Systems. (2010). Website, accessed on 24-06-2010. (`http://www.a123systems.com/a123/technology/power`)

Amelink, M. H. J. (2002). *Visual control augmentation by presenting energy management information in the primary flight display, an ecological approach.* Unpublished MSc thesis, Faculty of Aerospace Engineering, Delft University of Technology.

Amelink, M. H. J., Mulder, M., & van Paassen, M. M. (2008, March). Designing for human-automation interaction: Abstraction-sophistication analysis for UAV control. In *Proceedings of the International Multiconference of Engineers and Computer Scientists (IMECS).* Hong Kong.

Amelink, M. H. J., Mulder, M., van Paassen, M. M., & Flach, J. M. (2003a). Applying the abstraction hierarchy to the aircraft manual control task. In *Proceedings of the* 12^{th}*International Symposium on Aviation Psychology* (pp. 42–47). Dayton (OH).

Amelink, M. H. J., Mulder, M., van Paassen, M. M., & Flach, J. M. (2005a, April). Cognitive systems engineering approach to shared situation awareness for unmanned aerial vehicles. In *Proceeding of the* 13^{th} *International Symposium on Aviation Psychology* (pp. 7–13). Oklahoma City, Oklahoma, USA.

Amelink, M. H. J., Mulder, M., van Paassen, M. M., & Flach, J. M. (2005b). Theoretical foundations for a total energy-based perspective flight path display. *The International Journal of Aviation Psychology*, *15*(3), 205–

231.

Amelink, M. H. J., Mulder, M., van Paassen, M. M., Lintern, G., & Solodilova-Whiteley, I. (2006, September). Models for automation: Learning from autopilot design. In F. Reuzeau, K. Corker, & G. Boy (Eds.), *Proceedings of the International Conference on Human – Computer Interaction (HCI-Aero)* (pp. 160–163). Seattle, Washington, USA: Cépaduès-Éditions, Toulouse, France.

Amelink, M. H. J., van Paassen, M. M., & Mulder, M. (2009, April 27–30). Example of work domain analysis applied to total energy control system. In *Proceeding of the* 15th *International Symposium on Aviation Psychology* (pp. 479–484). Dayton OH.

Amelink, M. H. J., van Paassen, M. M., & Mulder, M. (2006, March). Actor-agent collaborations: Abstractions for interfacing operators and UAVs. In *Proceedings of Cognitive Systems with Interactive Sensors (COGIS'06).* Paris, France.

Amelink, M. H. J., van Paassen, M. M., Mulder, M., & Flach, J. M. (2003b). Total energy-based perspective flight–path display for aircraft guidance along complex approach trajectories. In *Proceedings of the* 12th*International Symposium on Aviation Psychology.*

Ashby, W. R. (1956). *Introduction to cybernetics*. Chapman & Hall, London.

Bainbridge, L. (1982). New technology and human error. In J. Rasmussen, K. Duncan, & J. Neplat (Eds.), (chap. Ironies of automation). New York: John Wiley & Sons.

Bernstein, N. (1967). *Coordination and regulation of movement*. New York: Pergamon.

Borst, C., Suijkerbuijk, H. C. H., Mulder, M., & van Paassen, M. M. (2006, October). Work domain analysis for terrain awareness. *The International Journal of Aviation Psychology*, *16*(4), 375–400.

Brockhaus, R. (1994). *Flugregelung* [Flight control]. Berlin: Springer–Verlag.

Brook, T. V. (2009, June). *More training on UAVs than bombers, fighters.* Website, accessed on 24–02–2010. (`http://www.airforcetimes.com/news/2009/06/gns_airforce_uav_061609w/`)

Bruce, K. R. (1987). *NASA B737 Flight test results of the total energy control system* (Tech. Rep.). NASA.

Burns, C. M., & Hajdukiewicz, J. R. (2004). *Ecological interface design*. CRC Press.

Castro, L. N. G., & Pritchett, A. (2005). Work domain analysis for improvement of uninhabited aerial vehicle (UAV) operations. In E. J. Bass (Ed.), *Proccedigns of the 2005 Systems and Information Engineering Design Symposium.*

Catton, L., Starr, A., Noyes, J. M., Fisher, A., & Price, T. (2007). Designing energy display formats for civil aircraft: Reply to Amelink, Mulder, van Paassen, and Flach. *The International Journal of Aviation Psychology*, *17*(1), 31–40.

Crouch, T. (1989). *The Bishop's boys*. Norton.

Dennett, D. C. (1971). Intentional systems. *Journal of Philosophy*, *LXVIII*(4), 87–106.

Dennett, D. C. (1981). Mind design. In J. Haugeland (Ed.), (chap. Intentional systems). Montgomery, Vermont: Bradford Books.

Dinadis, N., & Vicente, K. J. (1999). Designing functional visualizations for aircraft systems status displays. *The International Journal of Aviation Psychology*, *9*(3), 241–269.

Edmonds, B. (1999). The evolution of complexity. In F. Heylighen & D. Aerts (Eds.), (chap. What is Complexity? - The philosophy of complexity per se with application to some examples in evolution). Kluwer, Dordrecht.

Edwards, E. (1972). Man and machine: Systems for safety. In *Proceeding of British Airline Pilots Associations Technical Symposium* (pp. 21–36). London.

EMAV09. (2009, September). *Description of outdoor competitions.* Website, accessed on 20-08-2010. (`http://www.emav09.org`)

Endsley, M., & Kaber, D. B. (1999). Level of automation effects on performance, situation awareness and workload in a dynamic control task. *Ergonomics*, *12*, 101–107.

Endsley, M. R. (1995). Towards a theory of situation awareness in dynamic systems. *Human Factors*, *37*(1), 32–64.

Flach, J. M., Hagen, B. A., & Larish, J. F. (1992). Active regulation of altitude as a function of optical texture. *Perception & Psychophysics*, *51*(6), 557–568.

Flach, J. M., Patrick, P. F., Amelink, M. H. J., van Paassen, M. M., & Mulder, M. (2003). The handbook of cognitive task design. In E. Hollnagel (Ed.), (chap. A Search for Meaning: A Case Study of the Approach–to–landing). Lawrence and Erlbaum Associates.

Gibson, J. J. (1950). *The perception of the visual world.* Boston, MA: Houghton Mifflin.

Gibson, J. J. (1986). *The ecological approach to visual perception*. Hillsdale, NJ:

Lawrence Erlbaum Associates. (Original work published in 1979)

Gibson, J. J., & Crooks, L. E. (1938). A theoretical field–analysis of automobile–driving. *American Journal of Psychology*, *51*, 453–471.

Gladstone, B. (2000). *Performance racing trim* (sixth edition ed.). North U Performance Racing Seminars.

Gladstone, B. (2006). *Performance racing tactics* (fourth edition ed.). North U Performance Racing Seminars.

Henzinger, T. A., & Sifakis, J. (2007, October). The discipline of embedded systems design. *Computer*, 32–40.

Hollnagel, E., & Woods, D. D. (2005). *Joint cognitive systems: Foundations of cognitive systems engineering*.

Iacob, S. M., Nieuwenhuis, C. H. M., Wijngaards, N. J. E., Pavlin, G., & van Veelen, J. B. (2009). Actor–agent communities: Design approaches. In *Proceedings of th 8th International Conference on Interaction Design and Children (IDC).*

Kruit, J. D., Mulder, M., Amelink, M. H. J., & van Paassen, M. M. (2005). Design of a rally driver support system using ecological interface design principles. In *Proceedings of IEEE-SMC International Conference on Systems, Man and Cybernetics.* Hawaii.

Lambregts, A. A. (1983a, October). Integrated system design for flight and propulsion control using total energy priciples. In *AIAA Aircraft Design, Systems and Technology Meeting.* Fort Worth, TX, USA. (AIAA-83-2561)

Lambregts, A. A. (1983b, August). Vertical flight path and speed control autopilot design using total energy principles. In *Proceedings of the AIAA Guidance and Control Conference.* Gatlinburg, TN. (AIAA-1983-2239)

Lambregts, A. A. (1996). Automatic flight controls concepts and methods. In *NVvL annual report.*

Lambregts, A. A. (2009). Personal communication.

Langewiesche, W. (1944). *Stick and rudder: An explanation of the art of flying*. McGraw-Hill Inc. (renewed 1972 by Wolfgang Langewiesche)

Lind, M. (2003). Making sense of the abstraction hierarchy in the power plant domain. *Cognition Technology & Work*, *5*(2), 67–81.

Mackworth, N. H. (1950). *Researches on the measurement of human performance*. H. M. Stationary Office (London).

Marr, D. (1982). *Vision: A computational investigation into the human representation and processing of visual information*.

Mesarovic, M. D., Macko, D., & Takahara, Y. (1970). *Theory of hierarchical multilevel systems*. Academic Press, New York.

Militello, L. G., Dominguez, C. O., Lintern, G., & Klein, G. (2009). The role of cognitive systems engineering in the systems engineering design process. *Systems Engineering*.

Mulder, M. (1999). *Cybernetics of tunnel–in–the–sky displays*. Unpublished doctoral dissertation, Faculty of Aerospace Engineering, Delft University of Technology.

Mulder, M. (2003). An information–centered analysis of the tunnel–in–the–sky display, part one: Straight tunnel trajectories. *The International Journal of Aviation Psychology*, *31*(1), 49–72.

Mulder, M. (2007). *Haptic gas pedal feedback for active car-following support*. Unpublished doctoral dissertation, Delft University of Technology.

Mulder, M. (2009). From safety to where... ? *Inaugural speech on December* 16^{th}.

Newman, R. L. (1995). *Head–up displays: Designing the way ahead*. Ashgate publishing Limited.

Norman, D. A. (1990). The 'problem' with automation: Inapropriate feedback and interaction, not 'over–automation'. In *Philosophical transactions of the Royal Society of London* (Vol. B, pp. 585–593).

Oomkens, W., Amelink, M. H. J., Mulder, M., & van Paassen, M. M. (2008). UAVs as aviators: Environment skills capability for UAVs. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics.* Singapore.

Parasuraman, R., Sheridan, T. B., & Wickens, C. (2000, May). A model for types and levels of human interaction with automation. In *IEEE Transactions on Systems, Man, and Cybernetics – Part A* (Vol. 30).

Pincus, W. (2009, August). *Air force training more pilots for drones than for manned planes.* Website, accessed on 24-02-2010. (`http://www.washingtonpost.com/wp-dyn/content/article/2009/08/10/AR2009081002712.html`)

Pirsig, R. M. (1974). *Zen and the art of motorcycle maintenance*. William Morrow & Company.

Rasmussen, J. (1983). Skills, rules, and knowledge; signals, signs, and symbols, and other distinctions in human performance models. In *IEEE Transactions on Systems, Man, and Cybernetics (SMC)* (Vols. SMC–13, pp. 257–267).

Rasmussen, J. (1986). *Information processing and human-machine interaction*

(A. P. Sage, Ed.). New York: North-Holland.

Rasmussen, J. (1998, April). *Ecological interface design for complex systems: An example: SEAD – UAV system* (Tech. Rep. Nos. EOARD–Contract: F61708–97–W0211). HURECON.

Rasmussen, J., & Lind, M. (1981). Coping with complexity. In *Proceedings of European Annual Manual Conference on Human Decision and Manual Control.* Delft.

Rasmussen, J., Pejtersen, A. M., & Goodstein, L. P. (1994). *Cognitive systems engineering*. John Wiley & Sons Inc.

Sachs, G., & Sennes, U. (2001, August). Total energy related speed control for 3–dimensional guidance displays with predictor. In *Proceedings of the AIAA Guidance, Navigation and Control conference.* Montreal, Canada.

Sarter, N. B., Woods, D. D., & Billings, C. E. (1997). Handbook of human factors & ergonomics, second edition. In G. Salvendy (Ed.), (chap. Automation Surprises). Wiley.

Sheridan, T. B. (2000). HCI in supervisory control: Twelve dilemmas. In *Human error and system design and management* (pp. 1–12). Springer Berlin / Heidelberg.

Simon, H. A. (1981). *The siences of the artificial* (2nd ed.). MA: MIT press.

Stanton, N. A., Salmon, P., Jenkins, D., & Walker, G. (2010). *Human factors in the design and evaluation of central control room operations*. CRC Press.

The Dutch Safety Board. (2010, May). *Crashed during approach, Boeing 737-800, near Amsterdam Schiphol Airport, 25 Febuary 2009* (Tech. Rep.). (M2009LV0225_01)

Theunissen, E., & Rademaker, R. M. (2000). Augmentation for synthetic vision displays – an energy-based approach. In *Proceedings of the 2000 World Aviation Conference.*

Tufte, E. R. (1990). *Envisioning information.* Cheshire, CT: Graphics.

van Dam, S. B. J., Mulder, M., & van Paassen, M. M. (2004, October). Functional presentation of travel opportunities in flexible use airspace: an EID of an airborne conflict support tool. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics* (pp. 802–808). Den Haag.

van Paassen, M. M. (1995). New visualization techniques for industrial process control. In *Proceedings of the 6th IFAC/IFIP/IFORS/IEA Symposium on Analysis, Design, and Evaluation of Man–Machine systems* (pp. 457–462). Cambridge, MA: MIT Press.

van Paassen, M. M., Amelink, M. H. J., Borst, C., & van Dam, S. B. J. (2007).

The chicken, the egg, the workspace analysis, and the ecological interface. In *Proccedings of the* 14th *International Symposium on Aviation Psychology.*

van Paassen, M. M., & Mulder, M. (2004, Oktober). Functional and non-functional goals, design and usage of cognitive systems. In *IEEE International Conference on Systems, Man and Cybernetics.* The Hague.

van Paassen, M. M., Mulder, M., van Dam, S. B. J., & Amelink, M. H. J. (2005). 'Meaningful physics' or finding a system description suitable for ecological interface design. In *Proceedings of the* 13th *International Symposium on Aviation Psychology.*

Vicente, K. J. (1999). *Cognitive work analysis*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Vicente, K. J. (2002). Ecological interface design: Progress and challenges. *Human Factors*, *44*(1), 65–78.

Vicente, K. J., & Rasmussen, J. (1992, July/August). Ecological interface design: Theoretical foundations. *IEEE Transactions on Systems, Man and Cybernetics*, *22*(4), 589–606.

Wagter, C. de, & Amelink, M. H. J. (2007). Holiday 50 automatic vision. In *Proceedings of the International Micro Aerial Vehicle conference and flight competitions (IMAV07).* Toulouse, France.

Wikiquote. (2010). *Albert Einstein.* Website, accessed on 03–05–2010. (`http://en.wikiquote.org/wiki/Albert_Einstein`)

Woods, D. D. (1996). Automation and human performance: Theory and applications. In R. Parasuraman & M. Mouloua (Eds.), (pp. 3–17). Erlbaum.

Woods, D. D., & Cook, R. I. (1991, October). Nosocomial automation: Technology–induced complexity and human peformance. In *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics, 'Decision aiding for complex systems'* (Vol. 2, pp. 1279–1282). Charlottesville, VA, USA.

Woods, D. D., & Hollnagel, E. (2006). *Joint cognitive systems: Patterns in cognitive systems engineering*. CRC press, Taylor & Francis Group.

Yu, X., Lau, E., Vicente, K. J., & Carter, M. W. (2002). Toward theory–driven, quantitative performance measurement in ergonomics science: The abstraction hierarchy as a framework for data analysis. *Theoretical Issues in Ergonomics Science*(2), 124–142.

A

Example: the car as a transport system

This appendix demonstrates the added value of the ASA by extending an existing analysis, that was based on a single abstraction hierarchy, with the levels of control sophistication. An analysis of a car as a means to transport people is presented by Burns and Hajdukiewicz (2004) to explain Work Domain Analysis (WDA) with the abstraction hierarchy. Although it works well to explain the principles of the abstraction hierarchy, it is used here to show how the abstraction hierarchy alone fails to address the nested structure of this domain. This appendix demonstrates how the levels of control sophistication expand the analysis space compared to the abstraction hierarchy or Abstraction Decomposition Space (ADS). The expanded space allows a broader scope of the analysis and structuring of the knowledge according to the control problems encountered.

First, the original example is given. Then, the levels of control sophistication are motivated and introduced for this example. Finally, an overview of the ASA is presented and discussed by showing how a change in the system propagates through the levels of the analysis.

Original example

We start with an example abstraction hierarchy that is presented by Burns and Hajdukiewicz (2004) (p. 26). Figure A.1 shows their graphical ab-

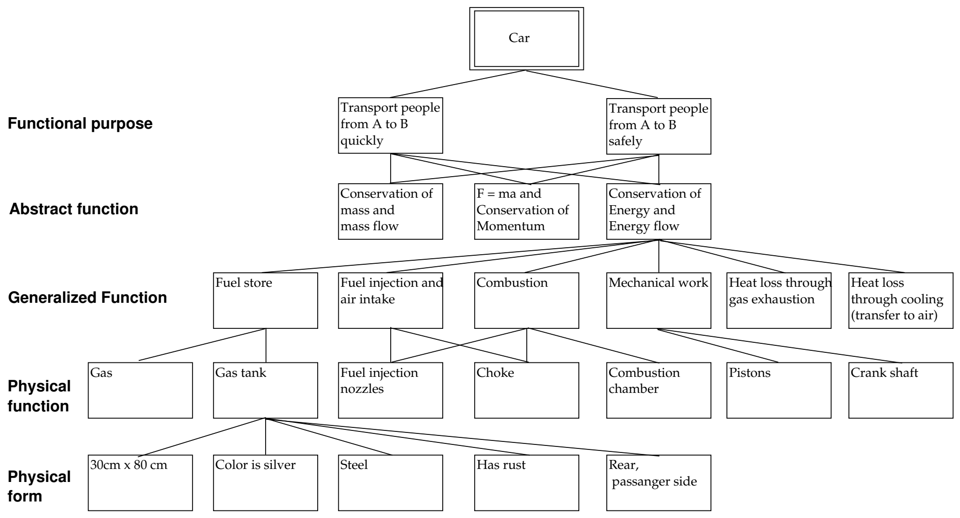

Car
Functional purpose
Transport people from A to B quickly
Transport people from A to B safely
Abstract function
Conservation of mass and mass flow
F = ma and Conservation of Momentum
Conservation of Energy and Energy flow
Generalized Function
Fuel store
Fuel injection and air intake
Combustion
Mechanical work
Heat loss through gas exhaustion
Heat loss through cooling (transfer to air)
Physical function
Gas
Gas tank
Fuel injection nozzles
Choke
Combustion chamber
Pistons
Crank shaft
Physical form
30cm x 80 cm
Color is silver
Steel
Has rust
Rear, passanger side

Figure A.1: Abstraction hierarchy analysis for a car as a transport system as presented by Burns et al.(2004)

straction hierarchy. Transporting people quickly and safely is defined as the functional purpose of the car. At the abstract function level, abstract physical laws are presented: the conservation of energy, the conservation of momentum (mass and mass flow), and Newton's second law: $F = M \cdot a$. In the analysis the choice is made to further detail 'Conservation of energy and energy flow' on the generalized function level where energy and energy flows are described using generalized functions (note the links between the 'conservation of energy box' and the level below).

At the generalized function level, descriptions are found of the functions that belong to engine and fuel system of the car: 'fuel store', 'fuel injection and air intake', 'combustion', 'mechanical work', and 'heat loss through gas exhaust and cooling'. At the physical function level some of the generalized functions are shown to be achieved by physical functions: 'gas', 'gas tank', 'fuel injection nozzles', 'choke', 'combustion chamber', 'pistons', and 'crank shaft'. Finally, at the physical form level the materials, positions and dimensions of the physical objects are represented.

The analysis links a number of concepts across levels of abstraction that mainly belong to engine and fuel system of a car. The abstract functions of the engine and fuel system can indeed be represented by the energy and mass flows. They are abstract physical models that provide descriptions of the constraints imposed on the transformation of chemical energy (fuel) to set the mass of the vehicle in motion. However, the link between the abstract function level and the functional purpose is *far fetched*. Mass, energy, and force representations cannot be shown to achieve transportation of people without additional representations. The control problem of using fuel as an energy source to achieve mechanical work is different from the control problem of transporting people from A to B.

More control functions are part of the work domain than that of the power conversion for locomotion. In the next sections the levels of control sophistication are applied in this analysis to create space to represent the control problems that need to be solved in order to see a car as a means for transportation.

Levels of control sophistication

By replacing the functional purpose of the original analysis: 'transport' with 'locomotion', the functional purpose better fits the analysis. Locomotion is achieved by energy conversion that is governed by the represented physical laws. Locomotion can provide transportation, but the analysis of transportation itself requires other insights.

Transport is represented by the outer–most level of control sophistication, and *locomotion* is represented at the inner–most level of control sophistication. Three additional levels of control sophistication are used to span the space between these levels: *controlled locomotion*, *driving*, and *navigation*. Figure A.2 shows the nested structure of the levels. Each level is exemplified with a number of functions and concepts that belong to that level, starting with the top level.

Transport – Transportation involves the flows of traffic in terms of volume and infrastructure capacity. The infrastructure can be represented in terms of road network capacity, and the availability of gas and service stations to sustain the traffic. The representation can show places that attract people and require a larger capacity of the infrastructure. Physically, these could be residential area's, business districts and entertainment centers. The focus at this level lies on the transportation system with its own representations for design, e.g., sustaining traffic flows.

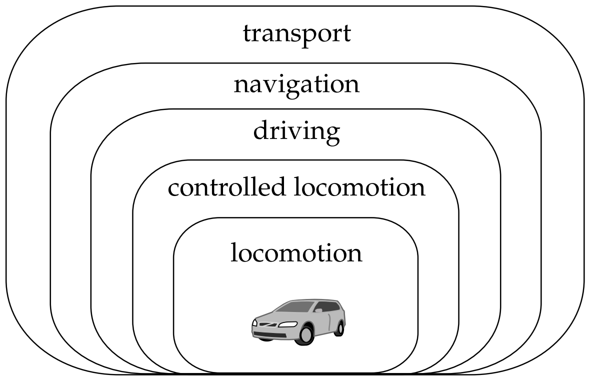


Figure A.2: Levels of control sophistication for car control.

Navigate – Zooming in from traffic flows to the individual car in the flow, this level represents the problem of getting to a destination. It involves the road network in terms of how places are connected and how it can be used to travel from a departure point to the destination. This encompasses the route planning and execution. Navigation systems that are often found in current cars, are good examples of automated systems that solve the navigation control problem. However, it solves the navigation control problem only. An end user is needed to decide what the destination is (at the transport level), and a driver is needed to drive the car, see below.

Driving – Driving deals with keeping the car on the road, in a lane, avoiding obstacles, and using the roads properly to implement the navigation solution. This level represents the roads in terms of the road, lanes, how to use them, speed limits and other regulations. It represents other road users, obstacles and how to deal with them properly. Driver–assist systems, e.g., auto braking systems that help keep distance, are being developed to automate the control problem of this level (Mulder, 2007). However, automation at this level is a challenge due to the correct sensing and processing of many variable and unpredictable outside influences.

Controlled locomotion – This level starts to have a tighter coupling with the car. The direction and speed of its locomotion are controlled through operation of the car with the aim to drive as intended. This level represents the control problem of providing locomotion with the desired speed and direction using the throttle and steering wheel. It also includes the road as a support surface, the tire–road contact, and the car as a means to provide locomotion. The cruise control system found in contemporary cars can keep the speed of a car constant. It is an example of an automated solution at this level of control sophistication.

Locomotion – The content of this level is similar to the abstraction hierarchy that Burns and Hajdukiewicz (2004) presented, with the exception that the functional purpose level is replaced by 'locomotion'. The car is represented as a machine to provide locomotion through energy conversion and setting mass in motion.

Mapping each level of control sophistication on a abstraction hierarchy results in the ASA. Figure A.3 shows a graphical representation of the ASA.

Discussion

Each control problem has its own abstract functions . For example, the transport level represents the traffic flows, not viewing the individual road user, the navigation level has representations for using the road network for reaching a destination, and the locomotion level has representations for energy conversion. By separating the control problems, and the representations specific to those control problems, clutter is reduced. The nested dependencies also become visible, supporting the tracking of how constraints propagate through the system.

For example, lets see how the introduction of the electric car could propagate through the described transportation system? The *locomotion level* is clearly impacted as the electric energy storage and electric motor replace the fuel storage and piston engine. At the *controlled locomotion level* representation may not need to change that much. The control functions of an electric car will be very similar to that of a fossil fuel car. Likewise, the nature of driving and navigation will not be significantly affected. However, at the *transport level* the infrastructure is significantly impacted. Recharging an electric car is significantly different than refueling a fossil fuel car. It requires recharge stations and takes a long time compared to refueling fossil fuel cars, due to battery characteristics. Support at the infrastructure level is the bottleneck for the electric car to become a successful part of the transportation system.

Represented at the *locomotion level*, batteries are subject to rapid development. New technologies such as $LiFePO_4$ batteries allow a charge current of more than 3 times the battery's capacity, thus allowing a full recharge time

	levels of control sophistication				
levels of abstraction	transport	navigation	driving	controlled locomotion	locomotion
physical function	Business districts, resitential areas, entertainment centers. Gas stations.	Location of home, location of work. All roads, how they are connected, which direction they can be driven on, average speed they provide.	Road: lanes and traffic lights, traffic signs. Other road users: pedestrians, bikes, cars, busses, trucks.	Road: layout (straight, curves, intersection), support surface, roughness, bumps Car: tires, tire–road contact speed, position	gas, gas tank, choke, combustion chamber, pistons, crank shaft, fuel injection nozzles.
generalized function	Places of interest that attract road users. Infrastructure. $\boxed{\text{navigating car}}$	Point of departure Point of arrival Roads as a route to destination $\boxed{\text{driving car}}$	changing lanes cruising obstacle avoidance speed regulation $\boxed{\text{controlled vehicle}}$	accelerate decellerate stop left and right steering $\boxed{\text{vehicle platform}}$	Fuel store, fuel injection and air intake, combustion, mechanical work, gas exhaustion. Seats and interior.
abstract function	Road network in terms of volume flows and capacity.	Road network in terms of how places of interest are connected and can be reached by the individual car.	Roads in terms of lanes and other traffic, speed and desired place on the road. Traffic regulations	Processed that control the vehicle. Stearing, throttle control, shifting gears.	Convertion of chemical energy, to mechanical work. Laws of conservation of energy, and conservation of momentum.
functional purpose	Transport of people: volume, safety, efficiency	Ability to reach destination using roads network.	Control over position on the road, obeying regulations, proper usage of roads.	Control over speed and direction	Car – locomotion

Figure A.3: The ASA for the car as part of a transportation system, spanning levels of control sophistication from transport to the locomotion level.

of less than 20 minutes (A123 Systems, 2010) This is still longer than a refuel but brings the recharge within waiting time and bring the electric car closer to a plug–in replacement. These properties propagate all the way up to the *transport level* to become part of the representation of infrastructure: the infrastructure of recharge stations will depends on the action radius of the cars, the recharge time and needed capacity.

By only viewing how new technological constraints propagate *up* the ASA, the success of the electric car is measured against our current way of using a fossil fuel car in our current infrastructure. Perhaps battery technology development will provide the breakthrough needed to allow the electric car to be used in the same way as a fossil fuel car. Or, should the *transport level* be allowed to set new requirements on the lower levels of control sophistication? This could lead to planning recharges in advance instead of near on demand refueling as is the case in our current infrastructure – a gas station always seems close by. Can trips, recharges, agendas and meals be planned in advance to combine a recharge with a lunch? Can a partial recharge be planned during a relaxed schedule to allow the driver to hurry along later in the day during a busy part of the agenda? Can a driver assist system advise the driver to adapt his driving and control style to extend the endurance and more efficiently plan recharges during a long trip? By introducing technologies at the lower levels of control sophistication, the constraints that propagate up to the *transport level* may be altered in such a way that the electric car more quickly becomes a viable alternative to fuel powered cars.

These suggestions illustrate the bi–directional constraint propagation across levels of control sophistication. They can be placed at their corresponding locations in the ASA, making the ASA a framework for exploration and visualization. Compared to the starting point of Figure A.1, the introduction of the levels of control sophistication have expanded the space for problem representation. The scope of the analysis now actually encompasses the car as a transportation system what the original analysis did not achieve. The larger space for analysis has allowed an enriched description of the different problems linking the car to the transportation system is forms with the infrastructure. Additionally, new questions can be asked and new constraints, and new requirements can be tracked through the ASA representation.

List of abbreviations

ADS	Abstraction Decomposition Space
AFMS	Advanced Flight Management System
AH	Abstraction Hierarchy
ASA	Abstraction–Sophistication Analysis
ATC	Air Traffic Control
CSE	Cognitive Systems Engineering
CWA	Cognitive Work Analysis
EAD	Ecological Automation Design
EATIS	Energy Augmented Tunnel In the Sky (display)
EID	Ecological Interface Design
FCS	Flight Control System
FG&C	Flight Guidance and Control
FMS	Flight Management System
FPV	Flight Path Vector
GCS	Ground Control Station
HMI	Human Machine Interface
HUD	Head Up Display
JCS	Joint Cognitive System
LEP	Line of Equal Position
MAV	Micro Aerial Vehicle
MIMO	Multiple Input–Multiple Output
SA	Situation Awareness

SEAD	Suppression of Enemy Air Defense
SISO	Single Input–Single Output
TECS	Total Energy Control System
TERP	Total Energy Reference Profile
UAV	Unmanned Aerial Vehicle, also Uninhibited Air Vehicles
VMG	Velocity Made Good
WDA	Work Domain Analysis

Samenvatting

Ecologisch Ontwerpen van Automatisering, Uitbreiding op Werk Domein Analyse

In hoge risico domeinen zoals de luchtvaart, geneeskunde en de besturing van kerncentrales heeft automatisering nieuwe mogelijkheden geïntroduceerd, de efficiëntie verbeterd en de veiligheid verhoogd. Echter, automatisering vergroot ook het aantal koppelingen in een systeem dat de complexiteit onbedoeld kan verhogen voor de menselijke operator. De automatisering van een systeem verandert het werkdomein van de menselijke operator, en zijn rol verandert van het besturen van de kernprocessen naar het overzien van de geautomatiseerde processen. De complexiteit van de automatisering en het gebrek aan ondersteuning kan de moeilijkheidsgraad van de controle taak groter maken dan hij hoeft te zijn, en daarmee de veiligheid, productiviteit en efficiëntie beperken. Om de door de automatisering geïntroduceerde complexiteit aan te pakken en te kunnen beperken werd de *ecologische benadering* voor automatiseringsontwerp genomen. Deze benadering richt zich op de relatie tussen de mens en zijn werkdomein, dat bestaat uit zijn omgeving en het systeem dat hij bestuurt. Het onderzoeksdoel was om uit te zoeken hoe de ecologische benadering kon worden gebruikt om de menselijke operator te ondersteunen bij het besturen van geautomatiseerde processen. Ecologisch Ontwerp van Automatisering (EAD) werd geformuleerd op basis van het Ecologisch Interface Design (EID) principe.

EID werd omgevormd door de automatiseringscomponent uit het werkdomein te halen en dezelfde onderliggende vragen te stellen over de raakvlakken tussen de automatisering, het werkdomein, en de mens: "hoe kan de complexiteit van het werkdomein worden weergegeven?". De conceptuele *gedeelde domein representatie* werd gedefinieerd om te visualiseren hoe de ogenschijnlijke complexiteit van een systeem kan worden teruggebracht wanneer de operator en de automatisering dezelfde afbeelding gebruiken van de begrenzingen in het werkdomein voor de besturing. Als onderdeel van de *ecologische benadering* werd werkdomein analyse (WDA) gebruikt om de begrenzingen in een werkdomein te analyseren en weer te geven. Echter, WDA is als methode nog niet volledig ontwikkeld en heeft last van conceptuele en methodologische zwakten. Het onderzoek richt zich daarom op de verdere ontwikkeling en uitbreiding van WDA om geautomatiseerde processen weer te kunnen geven. Vier studies werden uitgevoerd, en elke studie bracht nieuwe inzichten voort in de toepassing en uitbreiding van WDA.

In de eerste studie werd EID toegepast op het ontwerp van het Energy Augmented Tunnel In the Sky Display. Dit display werd ontworpen om de piloot te ondersteunen bij de naderingsvlucht voor de landing door middel van de weergave van informatie voor energiemanagement. WDA onthulde verschillende representaties voor de besturing van de vliegtuigdynamica waaronder de energie koppeling tussen het verticale vliegpad en de snelheid als een belangrijk tussenliggend besturingsdoel. Na de analyse volgde een creatief ontwerpproces dat resulteerde in een vernieuwend display waarin de energiemanagement informatie volledig en grafisch geïntegreerd is met het tunnel-in-the-sky display. Op basis van evaluatie verwachten we dat piloten hun stuurstrategiën veranderen door een beter bewustzijn van de energie toestand met mogelijke positieve effecten op prestaties en werklast.

In de tweede studie werd het bestaande Total Energy Control System (TECS) geanalyseerd met WDA. TECS is een automatisch stuursysteem voor dezelfde stuurtaak die werd behandeld in de eerste studie. Het originele ontwerp van TECS is gebaseerd op een doortastende analyse van de fundamentele fysica van het vliegen dat de energiemanagement principes onthulde. In deze studie werd het ontwerp van TECS op de abstractie hiërarchie afgebeeld om zo de rol van de energiemanagement principes weer te geven als onderdeel van het hele geautomatiseerde systeem. De analyse slaagde erin om de begrenzingen die door de automatisering werden geïntroduceerd af te beelden en de redenen voor bepaalde ontwerpkeuzes uit te leggen. Echter, veel ontwerp kenmerken werden in dezelfde analyseruimte weergegeven. Hierdoor werden de energieprincipes vertroebeld. De nadruk kon op energiemanagement worden gelegd door de toepassing van aggregatie maar dat ging gepaard met het verlies van informatie. Om alle informatie over TECS af te kunnen beelden zonder vertroebeling, werden de niveaus van control sophistication geïntroduceerd. Op elk niveau werd de abstractie hiërarchie toegepast wat resulteerde in de Abstraction–Sophistication Analysis (ASA). De analyse en afbeelding van TECS was niet eenduidig. Het betrof een zoektocht naar de interpretatie van de niveaus van de abstractie hiërarchie en hoe de means–ends relatie effectief gebruikt kon worden samen met aggregatie om een betekenisvolle analyse uit te voeren. De analyse liet zien dat geautomatiseerde processen in WDA kunnen worden weergegeven.

In de derde studie werd de ASA gebruikt om het ontwerp van *SmartUAV* te begeleiden. SmartUAV is een mini–UAV systeem dat vanaf de grond af werd ontworpen en gebouwd. Dit systeem kan meerdere kleine UAVs vanaf een laptop computer besturen. Door SmartUAV te ontwerpen en te bouwen hebben we ervaring uit eerste hand opgedaan met hoe WDA, de ASA in het bijzonder, hielp om de begrenzingen van een geautomatiseerd systeem tijdens het ontwerp af te beelden en de genestelde structuur weer te geven. Beginnend op het laagste niveau van control sophistication (vliegend platform), maakte de toevoeging van elk hogere niveau het mogelijk om een groter deel van het complete werkdomein mee te nemen in de analyse, en de analyse te richten op abstractere aansturing van de UAVs (tot aan missie). Bovendien ondersteunt de ASA de visualisatie van hoe automatisering het werkdomein verandert, dus hoe geautomatiseerde processen op lage niveaus van control sophistication de processen op hoge niveaus beïnvloeden. Deze studie toonde aan dat de ASA een veel grotere probleem ruimte kon weergeven dan de originele WDA. De ASA voorziet een systematische manier om onderscheid te maken tussen de abstractie van het besturingsprobleem (control sophistication) en abstractie van de elementen behorende tot een besturingsprobleem (abstractie hiërarchie).

De vierde studie behandelt de analyse van een sterk gestructureerd domein waarin automatisering een minimale rol speelt: wedstrijdzeilen. Deze studie bracht een duidelijker beeld voort van de genestelde structuur die inherent is aan het werkdomein in tegenstelling tot de genestelde structuur van de automatisering zoals we vonden bij TECS en SmartUAV. De genestelde structuur bij het wedstrijdzeilen was het resultaat van de evolutie van dit domein en de mogelijkheden van de uitrusting, de zeilers, en de reglementen. Bovendien is er aangetoond dat de prestaties van de bemanning in de analyses kon worden meegenomen wat niet mogelijk was met een niet genestelde WDA. Dit was van belang omdat de menselijke prestaties de basis vormen voor het behalen van doelen op hogere niveaus van control sophistication zoals: bootsnelheid, tactiek en strategie.

De vier studies brachten inzichten in WDA voort en hebben geleid tot de uitbreiding van de abstractie hiërarchie met de niveaus van control sophistication om de genestelde structuur van de werkdomeinen weer te kunnen geven. Door het uitvoeren van de analyses is er uitgebreide ervaring opgedaan met de abstractie hiërarchie wat heeft geleid tot de vermindering van ambiguïteit en zwakten van de methode. We vonden dat de abstractie hiërarchie gebruikt kan worden voor het modeleren van de structuur van de kennis in een werkdomein, maar niet voor het modeleren van de kennis zelf. De abstractie hiërarchie is een raamwerk voor het structureren van kennis door verschillende representaties behorende tot een besturingsprobleem met elkaar te verbinden.

De ASA kan een groter werkdomein gestructureerd in beeld brengen dan de niet genestelde WDA die is gebaseerd op een enkele abstractie hiërarchie. Tevens kon met behulp van de genestelde structuur automatisering gestructureerd aan de analyse worden toegevoegd tijdens de ontwerpfase. Weergave met de ASA reikt van fysiek op de lage niveaus van control sophistication, bijvoorbeeld natuurwetten, tot niet-fysiek op de hoge niveaus, bijvoorbeeld wedstrijd reglementen. Het gebruik van de ASA heeft de complexiteit van het mini–UAV systeem niet inherent verlaagd, maar het ondersteunde het in kaart brengen van de elementen die bijdragen aan de complexiteit. Representatie van werkdomein begrenzingen werd gezien als een belangrijke manier om ogenschijnlijke complexiteit voor de operator te verminderen. Op gelijke wijze wordt de weegave van het werkdomein met behulp van de ASA gezien als een belangrijke manier om de ogenschijnlijke complexiteit voor de ontwerper te verminderen en uiteindelijk die van het geautomatiseerde systeem.

Acknowledgements

My fascination with how people design, build, use, interact with, and adapt themselves to technology started when I joined the Control & Simulation group at Aerospace where Max Mulder and René van Paassen became my MSc thesis advisors. An inspiring internship at Wright State University with John Flach further sparked my enthusiasm. The joy of working in this field with these people made me come back for my PhD research. Max and René became my PhD advisors and then my promotor and copromotor. While Max and René allowed me to bounce unconventional ideas around, my other promotor Bob Mulder always managed to keep me critical of my own work as an Ingenieur. I'm grateful for their input, questions and inspiration. I'm also grateful for the research position that Kees Nieuwenhuis gave me at Thales and the D-CIS Lab, and the tremendous freedom he gave me for this research. The diverse group of colleagues at the lab inspired lengthly discussion during lunch and made it a more than interesting place to work.

I thank Tony Lambreghts at the FAA for taking the time to provide me with detailed descriptions of TECS. Thanks also goes to Eddy van der Heijden at the D-CIS Lab and Christophe de Wagter at TUDelft for their technical expertise and collaborations on developing SmartUAV. Additionally, I thank David Mobach at D-CIS, Guido de Croon and Bart Remes at TUDelft for the time we spent making the EMAV09 event a success.

It has now been some time since I read a quote somewhere, and without remembering the reference I quote it here: *"the best way to fall out of love with a topic is to write a PhD thesis on it"*. In the beginning I took that quote lightly but towards the end there were times when this seemed more than true. During these times my parents Anke en Wout always motivated me and believed in what I set out to do. They were always there with their unconditional support, guidance and wisdom. My sister Karin often appeared a bit sceptical of my endeavor but her continuous reminder that there should be an end to this process has been very helpful. Towards finishing, my girlfriend Cécilia managed to keep my feet on the ground and repeatedly and successfully reminded me that there is more to life than a thesis. For the past months, my commitment to this work may have made her days look a bit greyer but she always made my days shine brighter.

Curriculum vitae

Matthijs Hendrik Jan Amelink was born in Brummen, in The Netherlands, on the 18th of December in 1974. He attended secondary school at Hong Kong Island School (1986–1988) and the Comenius College in Hilversum, where he obtained his VWO diploma in 1993.

In the same year, he enrolled as a student at Delft University of Technology and actively participated in student life and extracurricular activities. His special interest became the interaction between the human pilot and the aircraft. An inspiring internship with Prof. John Flach at Wright State University in 2001 set his course on the ecological approach to interface design. He obtained his MSc. degree in Aerospace engineering in 2002 at the Control & Simulation division with a thesis titled: *Visual Control Augmentation by presenting Energy Management Information in the Primary Flight Display*. The thesis was nominated by the Faculty of Aerospace Engineering for the NVvL (Dutch association for aerospace technology) award 2002.

After graduation, Matthijs worked as a consultant for Accenture but soon decided to pursue a PhD position. In 2004, he started working for Thales Nederland B.V. at the D-CIS lab where he carried out his PhD research. The research was conducted under supervision of the Control and Simulation division at the Faculty of Aerospace at the TUDelft. Currently Matthijs continues to work at Thales where he leads the development of mission management for the next generation of UAVs.

Stellingen behorend bij het proefschrift

Ecological Automation Design, Extending Work Domain Analysis

Delft, 18 oktober 2010
Matthijs H. J. Amelink

1. De niet-lineaire relatie tussen de cellen van de 'Abstraction–Decomposition Space' (ADS), zoals beschreven door Rasmussen (1986), maakt de representatie van de ADS als een twee-dimensionale matrix van weinig betekenis.
2. De circulaire means-ends relatie zoals Lind (2005) die beschrijft in de analyse van de warmtewisselaar, is het resultaat van het verwarren van besturingsproblemen. Deze kan worden voorkomen door het koelen van krachtcentrale en de koeling van de warmtewisselaar op twee niveaus van *control sophistication* te beschrijven.
3. Het evalueren van een ontwerp methode voor systemen kan alleen aan de hand van de evaluatie van de systemen die het resultaat zijn van de methode. Voordat er een voldoende groot aantal voorbeelden is geëvalueerd zal men moeten volstaan met de theoretische onderbouwing van de methode.
4. De kernwaarde van de toepassing van 'Work Domain Analysis' ligt in het goed gaan begrijpen van de besturingsproblemen in een werk domein, het in kaart brengen van de kennis die daarvoor nodig is, en het gestructureerd overdragen van die kennis aan bijvoorbeeld systeemontwerpers.
5. Wanneer de menselijke bestuurder van een systeem op basis van het ontwerp en certificering wordt ingezet om de veiligheid te waarborgen, kan elk ongeluk worden verweten aan menselijke fouten en niet aan het systeemontwerp.
6. Tijdens het wedstrijdzeilen vermijdt men op tactische gronden een pad dat sterk afwijkt van de rest van het veld omdat men liever slechts met een meter wint dan het risico neemt om als laatste te eindigen. De competitie om geldstromen in de wetenschap heeft dat zelfde effect en beperkt de vrijheid voor baanbrekend onderzoek.
7. Zonder het zo nodig fysiek kunnen beperken van vrijheden van mensen is het onmogelijk om wetten en regels te implementeren, ook als deze ertoe dienen de vrijheid te handhaven.
8. Oplossingsgericht denken heeft alleen zin wanneer het probleem in voldoende mate wordt begrepen.
9. Voorbeeld geven is niet de hoofdzaak in het beïnvloeden van anderen, het is de enige manier (Albert Schweitzer).
10. Voor sommige woorden met een abstracte betekenis is het beter om deze niet in concrete termen te beschrijven omdat anders de generieke betekenis ervan verdwijnt. Zo schreef Pirgsig (1974) over 'quality' expliciet zonder het te definiëren.

Deze stellingen worden opponeerbaar en verdedigbaar geacht en zijn als zodanig goedgekeurd door de promotoren, prof.dr.ir. J.A. Mulder en prof.dr.ir. M. Mulder.

Propositions accompanying the doctoral thesis

Ecological Automation Design, Extending Work Domain Analysis

Delft, 18 October 2010
Matthijs H. J. Amelink

1. The non-linear relationship between the cells of the 'Abstraction–Decomposition Space' (ADS), as described by Rasmussen (1986), makes the representation of the ADS as a two-dimensional matrix of little meaning.
2. The circular means–ends relationship as described by Lind (2005) for the analysis of the heat exchanger is the result of the confusion of control problems. This can be avoided by describing the cooling of the power plant and the cooling of the heat exchanger at two levels of *control sophistication*.
3. The evaluation of a system design methodology can only be evaluated by evaluating the systems that are the result of the methodology. Before a large enough number of examples is evaluated, the theoretical foundation of the method will have to be sufficient.
4. The core value of the application of 'Work Domain Analysis' lies in obtaining a good understanding of the control problems in a work domain, the mapping of the knowledge that is required, and communicating that knowledge in a structured way to, for example, system designers.
5. When the human operator is tasked with guaranteeing safe operation of a system based on design and certification, every accident can be ascribed to human error and not to the system design.
6. During sailboat racing, a team will avoid a path that deviates from the rest of the fleet on tactical grounds because they rather win by just one meter than take the risk of finishing last. The competition for funding for scientific research has the same effect and limits the freedom for groundbreaking research.
7. Without being able, when needed, to physically constrain the freedom of people it is impossible to implement laws and regulations, also when they serve to maintain freedom.
8. Solution oriented thinking is only meaningful when the problem has been adequately understood.
9. Example is not the main thing in influencing others, it is the only thing (Albert Schweitzer).
10. For some words with an abstract meaning, it is better to not describe them in concrete terms because otherwise they lose their generic meaning. Pirsig (1974) wrote about 'quality' explicitly without defining it.

These propositions are considered opposable and defendable and have been approved by the supervisors, prof.dr.ir. J.A. Mulder and prof.dr.ir. M. Mulder.