Introducing distributed networks to designers
Development of an educational software tool and a methodology for its evaluation.

Imara C. T. M. Speek
Introducing distributed networks to designers:
Development of an educational software tool
and a methodology for its evaluation.

MASTERC THESIS
Embedded Systems
Interactive Intelligence Group
Faculty of Electrical Engineering, Mathematics
and Computer Science

AUTHOR
Imara Cornelia Teresa Maria Speek
1506374

SUPERVISING CHAIR
Prof. Dr. Catholijn M. Jonker

SUPERVISING MENTOR
Tomasz Jaskiewicz

THESIS COMMITTEE MEMBERS
Marco Zuniga and
Ing. A.J.C. van der Helm

August 25, 2015
Abstract

Connectivity has become increasingly important in embedded devices. The available connectivity in everyday objects has inspired designers to develop methods and tools for designing interactive environments. These large-scale systems exist out of multiple embedded devices that interact with a user as if to be a single environmental entity. The interactive qualities of these environments are dependent on the network topology in which the embedded devices are structured. Current design methodologies for interactive environments use centralized networks, a topology where a central device controls all other devices. This methodology increases in complexity when the number of embedded devices and the size of the interactive environment increases.

The computer science community uses ad hoc distributed networks to accommodate scalability and modularity issues. Distributed networks use global-to-local programming to describe the complex global behaviour of an environment using simple local rules for each embedded device in the environment. The resulting emergent quality to these behaviours is similar to the behaviour seen in a flock of birds. The introduction of distributed networks to designers will enable a further exploration of the potential of these distributed networks in a different light than current ad hoc applications. Additionally, introducing the distributed network topology will solve scalability issues as seen in centralized networks and provide naturally emergent interaction patterns. However, distributed network topologies are either unknown or unappealing to designers due to their sole visibility in ad hoc applications such as distributed databases or peer-to-peer networks. These current applications do not inspire, nor motivate the exploration of the potential of distributed networks. The global-to-local programming paradigm is a threshold concept, and requires a transformed way of understanding the concept without which the learner cannot progress. The inability of designers to understand the global-to-local programming restricts them from designing inherently interactive distributed networks.

This thesis uses a multi-disciplinary approach to design and develop the educational software “Chopper” to introduce the interactive qualities of distributed networks to designers. Firstly, the user-centered design approach evaluates the design process Chopper has to support. Secondly, the technology-driven design approach extracts the interactive qualities in the programming of a distributed network. The assumption is made that these qualities will enable designers to understand and apply the global-to-local programming paradigm in their interaction designs for interactive environments. A methodology to measure the effect of methods and tools in the education of designers is designed to evaluate the effect of Chopper. The evaluation criteria are: the understanding of the subject matter, the creativity in the design process and the application of the subject matter in their designs. To measure the effect in something as inherently chaotic as the design process, a complementary research setup in the form of a design brief is recommended. An explorative comparative study using this methodology is performed to pilot test the methodology as well as evaluate the methods and techniques deployed within the preliminary version of Chopper.

Quantitative results suggest that Chopper both enhances the designer’s understanding of the interactive qualities in the programming of a distributed network as well as their ability to apply these qualities in their interaction designs. Qualitative results show that while designers remain to have trouble motivating the design for this alternative network topology, they are inspired and excited for further introductions.
A SPECIAL THANKS TO EVERYONE THAT HAS BEEN INVOLVED IN THE PROJECT, BOTH PROFESSIONALLY AND SOCIALLY.

# Contents

## 1. MASTER THESIS ASSIGNMENT

1. Research assignment .......................... 20  
   1.1. Research assumption ................. 20  
   1.1.2. Research objective .......... 21  
1.2. Research questions ..................... 25  
1.3. Related work ........................... 26  
1.4. Significance ............................ 28  
   1.4.1. Distributed networks .......... 30  
   1.4.2. Example applications ........ 34  
1.5. Thesis structure ......................... 38  
   1.5.1. Reading Instructions ........ 39  

## 2. BACKGROUND

2. The product designer ................. 44  
   2.1. The design process .......... 45  
   2.1.2. Design phases .............. 46  
   2.1.3. Sketching ..................... 48  
   2.1.4. The designer ............... 50  
2.2. The distributed network .......... 52  
   2.2.1. Redefinition ............... 52  
   2.2.2. Designing distributed networks .... 54  
   2.2.3. Interactive qualities ......... 58  
2.3. The tool ................................ 64
Assignment

Master thesis assignment

Objective
Research questions
Related work

Significance

Designers
Distributed networks
  Examples

Thesis structure

Reading instructions
Our fit-tracking devices are connected to our water bottles, which in turn notify our phones to notify us to drink another glass of water. This development of connected everyday objects, allowing them to send and receive data is known as the Internet of Things. Since the start of the era of ubiquitous computing (Weiser, 1996), design professionals started developing methods to design large-scale interactive networks. These current methods are particularly useful when designing room-sized interactions, but complexity arises as soon as the number of connected objects increases.

Arranging the interactive objects in a distributed network topology enables to scale room-sized interactions to building-sized, or even city-sized interactions. The interactive qualities in distributed networks display emergent behaviours similar to distributed interactions in nature, for example a flock of birds in migration. The global emergent behaviour through the interaction of smaller objects is alien and counter-intuitive to the interactions in a centrally controlled network. The distributed paradigm: global emergence through smaller local interactions, is seen as a threshold concept (Meyer, 2013) – a concept in need for a different way of understanding to progress in education.

Distributed network applications provide solutions for modern day issues such as the design and development of intelligent adaptive buildings (Jaskiewicz, 2013) (Ruzeli, 2010) and energy saving through intelligent distributed street lighting. Leading design professionals have made attempts at designing methodologies (Kuniavsky, 2010) to accommodate the design of ubiquitous computing through intelligent distributed street lighting, but none of these attempts have been successful yet.

The software package HiveKit is developed to provide non-specialists such as design students, easy access to distributed algorithms (Dulman, et al., 2014). HiveKit allows users to specify behaviors for the distributed networks and automatically generate and deploy this behaviour on a network. Despite of its good intentions, HiveKit fails as a design tool (Myers, et al., 2005). Its Graphical User Interface (GUI) is neither tailored for the development of distributed networks nor does it fit the skills and experience of design students.

An educational tool is necessary to introduce designers of all backgrounds - students in particular - to distributed networks. To bridge the gap between designers and the interactive qualities distributed networks have to offer, a multi-disciplinary approach is required. Subsequently, a methodology for evaluating educational tools and methods in the context of design is required to evaluate its effect and make recommendations for future works. The educational tool should eventually enable designers and non-specialists alike to design interactions for interactive environments inherent to the global-to-local programming paradigm, deployed on a distributed network.

1. Assignment

The software package HiveKit is developed to provide non-specialists such as design students, easy access to distributed algorithms (Dulman, et al., 2014). HiveKit allows users to specify behaviors for the distributed networks and automatically generate and deploy this behaviour on a network. Despite of its good intentions, HiveKit fails as a design tool (Myers, et al., 2005). Its Graphical User Interface (GUI) is neither tailored for the development of distributed networks nor does it fit the skills and experience of design students.

An educational tool is necessary to introduce designers of all backgrounds - students in particular - to distributed networks. To bridge the gap between designers and the interactive qualities distributed networks have to offer, a multi-disciplinary approach is required. Subsequently, a methodology for evaluating educational tools and methods in the context of design is required to evaluate its effect and make recommendations for future works. The educational tool should eventually enable designers and non-specialists alike to design interactions for interactive environments inherent to the global-to-local programming paradigm, deployed on a distributed network.
1.1. Research assignment

1.1.1. RESEARCH ASSUMPTION

The distributed computing paradigm is a threshold concept (Perkins, 2006), a conceptual building block that progresses the understanding of troublesome knowledge. Troublesome knowledge is knowledge that if difficult to acquire because it is counter-intuitive, alien or seemingly incoherent (Perkins, 1999). A well-known threshold concept is the third law of Newton: When a body exerts a force on a second body, the second body exerts a force in equal magnitude and opposite direction of the first body. Students need to understand that even when a fly sits on a table, the table exerts a force to compensate the fly's force. The global-to-local programming paradigm is another example of a concept that needs to be fully understood for a designer to progress in the subject of distributed networks.

Initial user research evaluated design students at the Delft University of Technology that were experienced in the prototyping of interaction designs. Five students received an introductory lecture with all the required knowledge about distributed networks. Merely a single student was able to produce an interaction design that was appropriate for a distributed network topology. All participants felt restricted in their creative process and mentioned their lack of understanding subsequent to the research. This lack of understanding and the lack of a design methodology for distributed networks was visible in their interaction designs. Resulting from the initial user research, the initial research assumption is made:

“if designers are able to explore and understand interaction designs for distributed networks, they will be able to produce creative applied interactive qualities for distributed networks in turn.”

1.1.1. RESEARCH OBJECTIVE

Introducing the interactive qualities in the programming of distributed networks to designers will provide solutions for complexity issues in the design of large-scale interactions, or interactive environments. It will also introduce a new design perspective to distributed networks: the design driven innovation of distributed networks might lead to a whole new range of applications.

Intelligent architecture, an intelligent architectural application of interactive environments, indicates the need for early testing (Jaskiewicz, 2013). A software tool enables early testing without having to deploy the entire system in hardware. The development of simulations within software is both cheap and can be delivered timely. Hence the first research objective of the thesis:

1. Develop an educational tool that enables design students to explore, understand and apply the interactive qualities of distributed networks.

A methodology for evaluating the effect of the educational tool is required to make recommendations for future works. A generalized methodology in the context of design can evaluate both the effect of Chopper and other subject specific educational tools and methods. Hence, the second research objective of the thesis:

2. Develop a methodology that evaluates the effect of educational tools and methods in the context of design.
STEP-WISE OBJECTIVES

Figure 1.3 illustrates the objectives through the ‘meta-levels’ in design. Each level presents a triad of the “designer makes product for user”, <designer, product, user>. The thesis takes on two levels of the figure. First the thesis presents the highlighted “tool designer makes educational tool for introducing distributed networks for product designer”, <tool designers, educational tool to introduce distributed networks, product designer> triad. This triad in turn enables the lower meta-level “product designer makes distributed interactive environment for user”, <product designer, distributed interactive environment, user> triad. Finally, the thesis takes on the role of the methodologist to design an evaluation methodology the tool designer can use to evaluate the tool: “methodologist makes evaluation methodology for educational methods in the context of design for tool designers, or curriculum designers for that matter”, <methodologist, evaluation methodology in the context of design, tool designer>.

The main objectives are extracted into several step-wise objectives:

1. Identify the key features of the design process,
2. Identify the interactive qualities of distributed networks,
3. Review current programming tools in design and computer-science fields on their ability to support the exploration of distributed networks,
4. Develop a list of requirements the educational software tool must fulfill,
5. Develop an evaluation methodology for the educational tool,
6. Design an educational software tool to introduce designers to distributed networks,
7. Develop a preliminary version of the tool,
8. Evaluate the preliminary version of the tool using the methodology,
9. Make recommendations for improvement of the tool and the methodology to enable future tool designers in the context of design.

Steps 1 - 4 illustrate the tool designer objectives, steps 5 - 6 illustrate the methodologist related objectives and finally steps 7 - 9 illustrate the tool designer’s objective using the resulting methodology.
1.2. Research questions

Following research questions are formulated to meet the research objectives and are answered in Chapter 8: Conclusions.

1. What process should be supported by educational software for introducing the interactive qualities of a distributed network to designers?

2. What interactive qualities of a distributed network need introducing to designers?

3. How can an educational software tool facilitate the introduction of distributed networks to designers?
   a. What methods or techniques in existing tools in the field of design and computer science can be adopted to introduce distributed networks to designers?
   b. What new methods, techniques and GUI elements can be adopted to introduce distributed networks to designers?

4. How can an educational software tool for introducing distributed networks to designers be evaluated?
1.3. Related work

HiveKit is a software package that gives "non-specialists easy access to distributed algorithms" (Dulman, et al., 2014). It takes a global-to-local programming approach to describe complex distributed behaviour using simple local rules. HiveKit is an open source visual programming software plug-in for Grasshopper—a programming plug-in for the 3D modeling software tool Rhino.

It allows the describing of simple local rules in a 3D simulated environment, which is particularly effective in the early testing phase of a distributed network. It supports the deployment of the described behaviour on a network of Arduino Due’s, preloaded with the HiveKit firmware.

Providing non-specialists easy access to distributed algorithms requires a full understanding of the non-specialists and the definition of easy access. The easy access requires the recognizability of using the Grasshopper and Rhino tools. These tools are commonly seen in Architecture and other studies that focus on 3D modeling.

The user interface of Grasshopper is counter intuitive and inappropriate for loop-based programming. The Grasshopper interface is not recognized by all non-specialists and not by all design students. The counter intuitive available loop functionality in the user interface of HiveKit introduces a steep learning curve for designers and decreases their motivation to choose distributed networks. It enables the user in the explorations of simulations of the behaviour as well as the actual deployment. It does not guide the user in the development of the behaviour and assumes initial knowledge and understanding about how to design an interaction for distributed networks, which the users in fact do not have yet.

HiveKit has made the first attempt at enabling non-specialists in the development of distributed networks, but has not focused on a proper introduction. The user interface prevents designers from understanding or creatively applying the available functionality. It is in dire need of a successor.
Notions such as the Internet of things, ubiquitous computing and ambient intelligence have increased the importance of connectivity in embedded devices (Weiser, 1996) due to the inherently large number of connected objects deployed in their applications. The increasing connectivity amongst embedded devices offers a variety of networking possibilities such as connecting everyday objects to enhance their intelligence. Design-driven innovation could unlock the hidden potential within this technology push of connected entities, transforming it into a technology epiphany (Verganti, 2009). Technology epiphanies communicate new meaning of a technology and have the potential to become market leaders. Examples of technology epiphanies are Apple¹ that started using MP3 technology for portable music players or Swatch² that started using quartz technology for watches.

This thesis presents a multi-disciplinary approach to introduce designers to distributed networks. It presents both a user-centered and a technology-centered design approach to design and develop an educational software tool. A preliminary version of the tool is made as to evaluate the design proposals derived from this approach. Chopper is the first tool that specifically enables designers in the design for distributed networks.

A methodology is setup to evaluate educational tools and methods specifically within the context of design. The methodology is meant for evaluating the effect of tools and methods in threshold concept contexts within design, but can be generally applied to evaluate tools and methods in other subjects. This means the tool can be applied to evaluate methods and techniques within a tool, to evaluate the overall success of the tool compared to current alternatives or to compare methods and tools to design an educational curriculum in the context of design.

A distributed network is characterized by autonomy, cooperation and communication: the distribution of authority, functionality and information respectively (Epema, 2014). In other words: connected independent entities communicate and cooperate to achieve an overall goal. The distributed network topology is particularly favorable for its scalability and modularity. Examples that use distributed networks are distributed databases such as the Internet, distributed operating systems such as peer-to-peer torrenting and distributed applications such as multi-player Online games.

The current development of distributed networks is limited to ad-hoc applications by distributed networks experts. The following sections illustrate its meaning for designers and presents examples and visionary applications that can be achieved using distributed networks.

For further details, please read Section 2.2. The distributed network.
Design students at the Industrial Design Engineering faculty of the Delft University of Technology are educated in basic electrical engineering skills and programming skills. They use prototyping platforms such as Arduino to prototype and test interactive design concepts that respond to user actions such as lamps, clocks or even coffee machines. These design students are able to prototype these interactions using a set of sensors, actuators and basic programming skills.

Interaction designs that scale to an environment, or "interactive environments", contain a multitude of interactive elements, or "embedded devices". Each embedded device is equipped with a set of sensors and actuators and is centrally controlled. This topology is similar to a single micro-controller reading and writing to a set of sensors and actuators, but now a single micro-controller reads and writes to a set of input and output micro-controllers.

This similarity makes a centralized network topology intuitive and recognizable for designers. However, as the number of elements in the topology increases, interactions become less responsive due to the number of connections the central node has to control. Worse, if the central node fails, the entire system fails.

Entities in a distributed network are modularly connected to neighboring entities without this single central controller. It does not contain a single point of failure and is favorable for its scalability. The distributed network distributes messages with a gossip-like communication: each node communicates its state to its neighbors, which in turn do the same. The global-to-local distribution of data has proven to be a threshold concept for design students in the initial user research.

A focus group of 5 design students received an introductory lecture about distributed networks with the required knowledge to start designing. The students were asked to produce interaction designs for a distributed network, followed by a short written exam. While the design students varied in their programming and interaction design skills, some qualitative interpretations can be made. Design students are able to produce interaction designs for standalone interactive products and are familiar with centralized networks and are were unfamiliar and inexperienced with distributed networks and any convention to design for distributed networks. Four out of five design students were unable to produce a single interaction design that uses interactive qualities seen in a distributed network. All design students confused the global-to-local programming with a centralized control. All students noted that the lack of overall control in a distributed network made it unappealing to design for a distributed network. This indicates the actual threshold design students have to overcome in order to start designing for and with distributed networks.
Street lighting has high potential for added intelligence due to the impact of its energy consumption. The large number of the same objects makes street lighting a perfect candidate for a distributed network. Connecting the lights in a distributed fashion increases the modularity and robustness of the system: each light only needs to be connected to the neighboring lights and does not impact the entire network when it fails. The global-to-local behaviour of the system supports the propagation of a local detection of a user. Each light can be initially turned off to save energy. When a person is detected, the message is spread throughout the network and illuminates the path in front of the person. This distribution enables energy saving without compromising a person’s safety when walking in the dark.

1.4.2. EXAMPLE APPLICATIONS

City infrastructure is another perfect example of an application that would prosper using a distributed network. Current city infrastructures are static—city squares upon which market stalls and fences are positioned when the context of the square changes. A distributed network allows for dynamic configuration of city segments inherent to their current interaction context. When people are walking across the square, a walking path is created, when groups are formed, or a market stall is positioned, a radius around it indicates its use. A distributed network can be used to guide queuing lines, divert people from accidents and indicate tram rails. The possibilities are endless.

Other applications that would prosper from the use of distributed networks are interactive entertainment applications such as interactive dance floors\(^1\). Currently, interactions are limited to individual tile responses or centrally controlled by a processing unit that controls every tile in the network. The latter enables more interesting interaction patterns than interacting with each individual tile, but is limited to room-sized interactions; larger systems become unresponsive. If the central controller has to control more entities, it will become slower and less robust.

A distributed network is able to support building-sized or even city-sized interactions. The emergent nature of the interactive qualities of a distributed network supports naturally evolving patterns and entities as seen in nature. For example, a user can initiate a ripple effect that propagates throughout the city.

---

1.5. Thesis structure

This chapter introduces the user in the user-centered design and the technology in the technology-driven project. It presents the research objective, the research questions and the significance of this thesis. Chapter 2 introduces the background research that shapes the thesis, the tools required for the tool designer to get started. Chapter 3 concludes the research with a list of requirements for the design of the Chopper. Subsequently it presents the design proposals and the final design of the tool. Chapter 4 presents both the tools and the outcome required for the methodologist: the evaluation methodology illustrated with the explorative comparative study to evaluate Chopper. Chapter 5 presents the results from this study while Chapter 6 and 7 evaluate and make recommendations respectively, both for Chopper, educational tools for introducing designers to distributed networks and the methodology. Finally chapter 8 concludes the thesis by reflecting on the research objectives, summarizing the answers on the research questions and a personal evaluation of the author on the thesis.

Inherent to the multi-disciplinary nature of this report are readers from multiple disciplines. Readers that are especially interested in the design of Chopper and recommendations for future work are recommended to read chapter 3. Design, chapter 5. Results and the Chopper specific recommendations in chapter 7.1. Chopper. Readers who are interested in the specific use of distributed networks within this thesis are recommended to read chapter 2. Background, specifically the section 2.2. The distributed network and 2.4. Conclusion. Readers with an interest in the methodology for evaluating educational tools and methods in design are recommended to read the chapter 4. Methodology, 5. Results and 7.2. Methodology for recommendations. Readers that are interested in the multi-disciplinary bridging of fields are encouraged to read chapter 2. Background, especially about other fields, chapter 6. Evaluation, 7. Recommendations and 8. Conclusions. Most of all, everyone is encouraged to read it all.
Background

The product designer
- The process
- The designer

The distributed networks
- Redefinition
- Design
- Interactive qualities

The tools
2. Background

This chapter introduces the background information required for the design of Chopper. Section 2.1. The product designer introduces Chopper’s user in this user-centered design approach: the product designer, further referred to as “the designer”. It illustrates the various goals a designer might have using Chopper dependent on the phase in the design process and their skills and expertise.

Section 2.2. The distributed network introduces the interactive qualities of a distributed network with emphasis on their importance for designers. It redefines the meaning of distributed networks specific for this thesis and introduces several distribution patterns.

Finally, section 2.3. The tool presents the evaluation heuristics for a good user interface. It presents other tools that have introduced new technology to designers or computer scientists and evaluates their value for Chopper.

As a result, chapter 3. Design presents a list of requirements Chopper must adhere to, based on the background information presented here.
This section introduces the designers and the chaotic process inherent to design. The design process is highly dynamic, iterative and each phase within this process has different design goals. These differences influence the use of an educational software. This section discusses the variability amongst product designers and introduces the different roles each type of designer has in the design process. Additionally, sketching, the archetypal activity of design is introduced, and its representation in code: sketching in code. The goals and individual roles of the designers are subsequently used to develop a list of requirements Chopper must adhere to.

The design process is the entire process leading up to a production ready design. It is inherently chaotic, iterative and takes many shapes. Generally, it starts with a quantitative divergence and ends with a convergence as is shown in Figure 2.4. Quantity is key in divergence, because “in order to have a good idea, one has to have many ideas” (Pauling) (Buxton, 2007). The divergence is important in order to innovate before converging to a final design.

Within the divergence and convergence, the iterative design process cycles through constant discoveries and developments that increase the knowledge of the designer. Each iteration in the design process leads to new discoveries and development that can be evaluated. This iterative incremental process is key to increasing quality in the designs.

“That squiggle of the design process” represents the chaotic convergent, divergent and iterative nature of the design process. It starts by defining the problem, followed by research, followed by concept generation and finally concludes in a design. Note that the process can go back and forth between phases. Within the design process, designers can have different design goals. These design goals are dependent on the design phase within the process.
2.1.2. DESIGN PHASES

The design process exists out of multiple phases inherent to the type of design process. Each phase of the process has different goals and influences the behaviour of the designer. The tool should adhere to the different goals and behaviors in the design phase that matches the introduction of distributed networks best.

The Interactive Technology Design course at the faculty of Industrial Design Engineering is a course in which product designers are introduced to the potential of embedded devices - products embedded with a micro-controller, sensors and actuators. Design students are challenged to creatively apply interactive qualities of embedded devices. This thesis challenges design students to creatively apply interactive qualities of a distributed network. The design phases within the Interactive Technology Design course are inherently similar and subsequently used as a guideline for the phases in the process of designing for distributed networks.

The course process is segmented in 5 predefined iteration phases:

1. Hacking the demo: adapting and modifying an example to incorporate an aspect of the design brief.
2. Autonomy: creating a first standalone prototype of the solution without experience or knowledge of the used platform, from scratch or by hacking examples.
3. Nut-cracking: evaluating the results followed by further development to a mature level. Demonstrating mastering of the technology by solving the hardest technological problem.
4. Users: users are involved to test an interactive prototype to observe any usability issues.
5. Integration: The final prototype is presented to the employer.

Inherent to the introduction of distributed network, focus is set on the hacking phase in the design process as design students are yet unexperienced with distributed networks. The autonomy and nut-cracking phase show important goals and behaviors that will be apparent once the designers are properly introduced to distributed networks. The following sections illustrate the different goals within the most relevant phases of the design process: hacking the demo, autonomy and nut-cracking.

**Figure 2.5.** The design process of designing interactive qualities

**HACKING THE DEMO**
Demos should be available to be able to hack demos. Inherent to easy hacking, modification tools are recognizable and fault-avoidant. Key features in the interaction with distributed networks are available for modification and self-explanatory. Modification of appearances, positioning and types of sensors and actuators allow for relatively straightforward incorporation of the design brief in the demo.

**AUTONOMY**
Functionality is built from scratch or incrementally on existing demos and can be reused amongst files. No knowledge or experience in the development of distributed networks is required. Without touching the global-to-local programming, inputs, outputs and local rules can be modified.

**NUT-CRACKING**
This phase calls for deeper understanding of the distribution patterns. Debug views enables the user to track the behaviour of the local nodes. The code is available to the user and allows for modification.
“Sketching is not only the archetypal activity of design, it has been thus for centuries” (Buxton, 2007)

2.1.3. SKETCHING

Sketching is used throughout the entire design process and is central in design thinking and learning (Buxton, 2007). Sketches are used to visualize and communicate ideas and concepts. Sketches are quick to make and can be provided when needed, making it an ideal tool to support the chaotic and iterative nature of the design process.

A designer sketches from what is known. Reading sketches enables the designer to see new perspective or insights previously unknown. These sketch iterations allow the designer to incrementally increase their knowledge and produce better designs as is shown in Figure 2.6.

Figure 2.7 presents the qualities of a sketch. These qualities are important when developing an educational tool in the context of design. Educational processes that are similar to sketches will have a higher recognizability and thus acceptance by designers.

Of course, sketches have their limitations. The process of sketching in code should have the following properties (Margolis, 2011):

- The ability to learn from mistakes
- The ability to freely experiment with examples
- The ability to test and iterate on code
- The ability to verify new or altered code on correctly compiled changes
- The ability to work in incremental steps and undo mistakes.

These properties should be taken into consideration when designing an educational tool that supports the explorative nature of programming related contexts such as distributed networks.
2.1.4. THE DESIGNER

There is no single designer. Each designer has specific skills, interests and experience that influences their goals during the design process. These properties are inherent to their design role within a design team. The different responsibilities in the teams as seen in the Interactive Technology Design course (van der Helm, et al., 2013) are shown in Figure 2.9.

<table>
<thead>
<tr>
<th>Role</th>
<th>Responsibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Designer</td>
<td>In charge of conceptual development</td>
</tr>
<tr>
<td>Maker</td>
<td>In charge of construction of mechanisms or electronics</td>
</tr>
<tr>
<td>Programmer</td>
<td>In charge of software for product behavior</td>
</tr>
<tr>
<td>Manager</td>
<td>Keeps an overview and holds group members to their responsibilities</td>
</tr>
<tr>
<td>Blogger</td>
<td>In charge of representation on blog</td>
</tr>
<tr>
<td>Director</td>
<td>In charge of communicating about the interactive concept</td>
</tr>
</tbody>
</table>

Each designer has a different main responsibility. These responsibilities influence the behaviour of the designers. Figure 2.10. Responsibilities of the most significant designers in the design process shows the main responsibilities of the most relevant designers in the process of designing for distributed networks.

DESIGNER
The designer focuses on the conceptual development of the interaction. The designer explores as many interaction designs with minimum effort and knowledge of the technology. The explored interactions are compared. A representation of the interaction in the real world is important. The designer ignores structural hardware and software issues to diverge as much as possible. The designer takes a user-centered approach and wants to explore many different use cases.

The designer is responsible for conceptual development in the hacking and autonomy phase and again when including the users.

MAKER
The maker is interested in the structure of the system and the deployment possibilities. The maker looks into the autonomous setup of the interactive entities as well as the connections to realize the topology. The maker explores the consequences of failures in the system and explores how to use the system once it’s deployed into the world.

The maker is mainly responsible for the integration of the prototype, but supervises the process during the hacking, autonomy and nut-cracking phases.

PROGRAMMER
The programmer focuses on the behavioral software that causes the interactive qualities of the system. The programmer wants to thoroughly understand the global-to-local programming and the distribution patterns that cause the autonomous behaviors in the examples. The programmer wants to step through the behaviour of the system to understand all aspects as well as have full access to the code.

The programmer is mainly responsible during the nut-cracking phase of the process, but supervises the hacking and autonomy phase. This thesis focuses on the hacking and autonomy phase of the design process. The programmer is therefore expected to be somewhat frustrated as programmers won’t be able to actually dig into the code when Chopper support merely hacking.

MANAGER, BLOGGER & DIRECTOR
The manager oversees the entire design process. The manager will look into tangible results – the behaviour of the system. The blogger and director communicate the functionality of the system to the outside world. These personas should be able to make screenshots and record any interactions.

These roles are global roles that don’t indicate skills and experiences as relevant as the designer, maker and programmer. These roles will therefore not be used for indicating skills and experience in this thesis.
2.2. The distributed network

“A distributed system is a collection of independent computers that appears to its users as a single coherent system” (Tanenbaum, 1994). This means that firstly, a distributed system consists of components (i.e. computers, sensor nodes, Arduinos, high-performance mainframes etc.) that are autonomous and act independently. Secondly, it means the users of the system interpret the network as a single global entity, similar to a flock of birds. Finally, it means that the autonomous components need to collaborate in some way.

The user is unaware of the communication that happens between components in a distributed network. Instead, users interact with a distributed network in a consistent and uniform way, independent from location and time. The independent components cause the distributed network to be relatively easy to expand or scale. Because of this there is no independent failure mode and users are often unaware of a failure or defect in the network.

Centrally controlling this networks is not recommended since messages will overload most parts of the network. Global-to-local programming is used to describe the globally complex behaviour in local rules (Dulman, et al., 2014). Global-to-local algorithms have the following characteristics:

1. No component has complete information about the global state of the network,
2. Each component makes decisions based solely on local information,
3. Failure of one component does not fail the algorithm,
4. There is no implicit assumption that a global clock exists.

A distributed embedded or pervasive system is characterized by smaller, mobile embedded devices connected wireless that are part of our surroundings and is inherently distributed (Tanenbaum, 1994). Example applications are electronic health monitoring systems that prevent people from being hospitalized and small sensor networks consisting of ten to hundreds or thousands of relatively small nodes.

2.2.1. REDEFINITION

Designers are experienced in the prototyping of interactive qualities for standalone embedded devices, similar to the nodes in a distributed pervasive system. Pervasive systems are characterized by the aware or unaware interaction with a user. Such a system offers the potential for user-centered interaction design. This thesis focuses on such pervasive systems as distributed networks of embedded devices, equipped with sensors and actuators.

Further references to distributed networks focus specifically on homogeneous distributed pervasive systems in which all smaller entities are the same. This eases the process of understanding global-to-local programming similar to systems as seen in nature: a flock of birds, a swarm of bees, a school of fish etc. Similarly, the embedded devices are not yet to be connected wireless, but maintain a wired connection and a wired power source to reduce complexity.
2.2.2 DESIGNING FOR DISTRIBUTED NETWORKS

This section focuses on the considerations for designing distributed networks as illustrated in Figure 2.12. It presents guidelines and pitfalls that are apparent in the process of designing interactions for distributed networks and the most valuable interactive qualities for designers to use.

A distributed network strives to be transparent—hide the fact that its processes and resources are physically distributed across multiple embedded devices from the user. An example of such transparency is seen in a school of sardines that swim as if to be a single entity as a defense mechanism to confuse and scare off sharks.

The user case is the product designer in this case. In order to design for distributed networks, the product designer has to be aware of the fact that the processes and resources are physically distributed. Only if the product designer is able to understand and apply this opaqueness, he will be able to design a transparent distributed network for the consumers.

![Figure 2.12](image1.png)

Figure 2.12: The goal of the product designer in the meta-levels of design (Stappers & Hoffman 2009) - slightly altered for highlights.

PITFALLS

Relevant properties of distributed networks are reliability, security, heterogeneity, and the topology of the network (Tanenbaum, 1994). However, when developing a distributed network application, there are some of the false assumptions and pitfalls that are made:

1. The network is reliable,
2. The network is homogeneous,
3. The topology does not change,
4. Latency is zero,
5. Bandwidth is infinite.

Reliable networks simply do not exist and thus achieving failure transparency is impossible (Tanenbaum, 1994). Similarly, latency and bandwidth problems can occur depending on the application deployed on the distributed network. Similar to the opaqueness of the network, these pitfalls need to be communicated properly to the product designers before the design is deployed in real life.

![Figure 2.13](image2.png)

Figure 2.13: Sardines portraying inherent global-to-local transparency to confuse a shark.
DESIGN GUIDELINES

While properties of a distributed network are strongly context dependent, some initial guidelines can be provided to (un-)motivate the design for distributed networks:

The disadvantages are mostly related to the lack of experience and conventions in the field of applying distributed networks. This thesis focuses on laying the foundation that will enable designers to design interactive qualities for distributed networks without the need for experience and conventions. Similarly, designers are encouraged to look for inherent distribution in their interaction designs for the distributed networks.

<table>
<thead>
<tr>
<th>ADVANTAGES</th>
<th>DISADVANTAGES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Costs – Simpler embedded devices often offer a better price/performance than mainframes</td>
<td>Not enough experience in designing, implementing and using distributed software.</td>
</tr>
<tr>
<td>High reliability – There is no single point of failure.</td>
<td>There are no conventions in operating systems, programming languages or applications.</td>
</tr>
<tr>
<td>Incremental growth – Modularity of the local components allow the network to scale in size.</td>
<td>There are no convention on how much the users of the network should know about the distribution.</td>
</tr>
<tr>
<td>Speed – On large scales, these is more computing power available than in a mainframe.</td>
<td>The Network can saturate or lose messages.</td>
</tr>
</tbody>
</table>

Inherent distribution - Sometimes it makes sense in context to make local decisions.

Security – Easy access also applies to secret data.

Figure 2.14 The advantages and disadvantages of designing for a distributed network

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>STANDALONE</th>
<th>CENTRALIZED</th>
<th>DISTRIBUTED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interactivity objects</td>
<td>Single</td>
<td>Limited</td>
<td>Highly scalable</td>
</tr>
<tr>
<td>Uniformity</td>
<td></td>
<td>Hetero/homogeneous</td>
<td>Homogenous</td>
</tr>
<tr>
<td>Interactive range</td>
<td>User-sized</td>
<td>Room-sized</td>
<td>Building-sized</td>
</tr>
<tr>
<td>Common state</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Single point failure</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

Figure 2.15 Guidelines for choosing an interactive structure
2.2.3. INTERACTIVE QUALITIES

This section introduces the interactive qualities of a distributed network that are responsible for the overall distribution patterns they offer. Key interactive qualities for interactions design of distributed networks are emphasized.

Nodes in a distributed network are first and foremost autonomous. They contain sensors, a set of behavioral rules and actuators. Each independent node thus has the capability to respond accordingly to a specific input – each node represents a homogeneous standalone interactive entity with added connectivity to form a network.

Global-to-local programming entails the emergence of global patterns, interactions and virtual appearances of larger entities derived from smaller local interactions – communication and collaboration – in the network. Each node communicates their state to neighboring nodes, which in turn do the same. If a single node changes state, this state distributes through the system, apparent in Figure 2.16.

The distribution of states can be independent or dependent on user interaction. Dependent interactions are further referenced as environmental interactions: an interaction the user initiates in an environment. Independent interactions are referenced as interactive environments: an environment that portrays emergent behaviour prior to any user interaction. These interactive environments can initiate state changes and distributions based on random factors, or environmental inputs such as temperature, light measures or walking patterns from memory. These interactions are not mutually exclusive and can be used together.

Data can distribute through the network in a variety of patterns. These patterns define the distribution of data and the interactive behaviour of the system. The following sections discuss some well-known distributed algorithms that may prove convenient for interaction designs.

Figure 2.16. Distribution of a state change through the network
The gradient algorithm distributes in a similar fashion as the local-to-global distribution. However, instead of propagating the same state throughout the network, each node changes its state to a gradient fade and propagates the new state to its neighbors as seen in Figure 2.18.

Each gradient algorithm has 3 settings that determine the overall behaviour:

1. The radius of the distribution – the number of steps the gradient travels from the triggering condition (in Figure 2.18 the radius is 3)
2. A maximum value – assumed at the triggered node (dark teal)
3. A minimum value – assumed at the node n steps away from the triggering condition (light blue)

The three properties determine the overall pattern of the gradient. When a single gradient algorithm is used, solely the radius and the mapping of the state value to an output value is important. However, when dealing with multiple distribution algorithms, the relations between these multiple values becomes of importance. These relations are determined with the minimum and maximum values.

Figure 2.19 illustrates the relation between 2 gradient patterns. The blue color coded gradient has a steps radius of 4, a maximum value of 4 and a minimum of 1. The user gradient, illustrated in gray has a steps radius of 3, a maximum value of 3 and a minimum of 1. While the user does not initially experience the gradient following him, it becomes apparent when the user interacts with the other pattern: the states are subtracted from one another output the color. This relation could have been weaker: when the color coded gradient has a maximum of 40 and a minimum of 10, the user has to work together with multiple users for instance to achieve the same results.
The token distribution pattern passes a single state through the network as seen in Figure 2.20. When a state change is initiated, the state propagates through the system, one step at a time. The specific propagation of this state change can be completely random, based on another “invisible” gradient pattern or an external input such as temperature. The token message either propagates forever or has a specific Time-To-Live (TTL), the length of steps before the message is destroyed.

Each token algorithm has 4 settings that determine its overall behaviour:
1. The maximum length of forwarding (TTL)
2. The minimum value of the token
3. The maximum value of the token
4. A forwarding protocol.

In order to apply a token algorithm in the interactive qualities of a distributed network, designers will need to understand the limitations of the forwarding protocols. Local forwarding is a lot easier than path finding within the network.

Many other distribution patterns can be deduced from the gradient and token patterns. The distribution of these patterns is determined by spatial and temporal properties in the distribution. Spatial properties are related to space: e.g. where the message comes from and where it should be propagated to as illustrated in Figure 2.24. Each node specifies a direction from which to receive messages and reacts accordingly. Rule 30 (Wolfram, 1983) is a cellular automata in which each node determines its state on the above nodes as can be seen in Figure 2.23.

Similarly, the distribution pattern can be dependent on temporal qualities, related to time. These temporal qualities are related to how many nodes need to message their state change before the current node changes state. As seen in the token distribution, this can either be a single node, or as seen in the gradient all neighboring nodes, or any number in between as illustrated in Figure 2.21.

Custom distribution patterns outside of cellular automata such as Rule 30, use a combination of both: spatio-temporal qualities to determine the global behaviour. Figure 2.22 displays a temporal initiated state change from a single neighboring node. The node changes its state accordingly and distributes the message based on spatial properties: the neighboring nodes opposite of the incoming message.
2.3. The tool

This section presents the heuristics that are used to evaluate a user interface, the guidelines for the development of the user interface of Chopper. Subsequently it presents other tools that have already been successful in the introduction of threshold concepts in the context of design and computer science. Methods and techniques of these tools are presented and evaluated on their ability to introduce distributed networks to designers and used as an inspiration for the design of Chopper.

2.3.1. USER INTERFACE

The educational software tool presents the interface between the product designer and the distributed networks technology. The interface is responsible for the communication of the interactive qualities of the distributed network so that a product designer will be able to understand and apply these qualities. The user interface should support the explorative, iterative nature of designers and visually connect the dots. The quality of a user interface are evaluated using the Nielsen heuristics (Nielsen, 1995).

These heuristics are evaluated in the specific context of an educational software tool that introduces distributed networks for designers and are used to setup the design requirements of the tool.

Ten Usability Heuristics by Jakob Nielsen

- **Visibility of system status**: Give the user appropriate feedback about what is going on.
- **Match between system and the real world**: Use real-world words, concepts and conventions familiar to the users (in a natural and logical order).
- **User control and freedom**: Support undo, redo and exit points to help users leave an unwanted state caused by mistakes.
- **Error prevention**: Prevent problems from occurring, prevent error prone conditions or check for them before users commit to the action.
- **Consistency and standards**: Follow platform conventions through consistent words, situations and actions.
- **Recognition rather than recall**: Make objects, actions, and options visible at the appropriate time to minimize users memory load and facilitate decisions.
- **Help and documentation**: Make necessary help and documentation easy to find and search focused.
- **Help users recognize, diagnose, and recover from errors**: Express error messages in plain language (no codes) to indicate the problem and suggest solutions.

![Ten usability heuristics (Nielsen, 1995)](image)
This section presents tools in the field of design and computer science that have been used to introduce technologies. It highlights the methods and techniques deployed for the introducing these technologies to designers or computer scientists. The evaluated tools are Eclipse, Processing, Scratch, NetLogo, Mindstorms, Unity and Max MSP.

2.3.2. SOFTWARE TOOLS

MULTI-MULTI-FUNCTIONAL (IDES)

Integrated Development Environments (IDES) such as Eclipse and Unity, are software applications designed to maximize the programmer’s productivity by providing tight-knit components in a single program with similar user interfaces. This reduces the configuration needed to piece multiple development utilities together. Eclipse includes a source code editor, build automation tools, a compiler, an interpreter and a debugger.

Unity is a cross-platform game creation system that includes a game engine and an IDE and provides an inspector, file hierarchy, project files while centralizes the scene, game and animator window.

These multi-multi-functional applications are convenient for experienced users, but has a steep learning curve. It supports the user in the entire design process – from hacking, to nut-cracking to full integration of the functionality. It is a very powerful tool, but requires a large community and set of tutorials for the user to get started.

BARE MINIMUM

Arduino is based on the open source Processing language and the IDE Processing. Processing is developed to get non-programmers started with programming in the communities of electronic arts, new media arts and visual design. The IDEs are different form the multi-multi-functional IDEs because they present a bare minimum of the functionality: a simplified syntax of the C and Java language respectively and hot-keys to compile and upload a program to the Arduino micro-controller. The bare minimum name is deducted from the example sketch Arduino provides presenting the bare minimum for a sketch to compile with errors.

Arduino is supported by a very large maker community that is active on Arduino forums and other forums such as Instructables. The Online community supports their users in the design and use of standalone interactive setups and produces many example projects to start with. While the tool itself minimized the functionality to get the user started, the programming remains difficult. Arduino provides a very large set of examples and the active community enables the user to overcome this initial difficulty and start making autonomous behaviors.

VISUALIZED TECH

NetLogo is a multi-agent system IDE that provides both a development environment and a separate simulation window. The agents in a multi-agent system are autonomous, locally tracked and decentralized, which enables the development of distributed networks of agents. NetLogo offers direct feedback and experience of the functionality. It presents alternative views inherent to the different goals in using the tool:

- Interface: a visual interface for exploration and visual experiencing of the behavioral code.
- Info: a textual reference that explains the example using the following headers:
  - What is it?
  - How to use it?
  - Things to notice.
  - Things to try.
  - NetLogo features.
  - Related models.
- Code: the agent based code for NetLogo that determines the behaviour.

The alternative views enable the user to explore the functionality inherent to their goals, it hides the detailed information from inexperienced users, but still enables fast traversing between all views.
BUILDING BLOCKS

Scratch\(^1\) and Mindstorms\(^2\) are programming environments specifically built for children to teach basic skills in programming. What they lack in freedom, they make up for in comprehensibility and effectiveness. Both tools use building blocks that represent pieces of code. The blocks have physical restrictions to guide the building of the entire program while avoiding faults. Different functionalities are color coded to display the sets of functions. Scratch uses recognizable syntax for the target group such as movements, appearance, events and control instead of using difficult terminology to direct itself to the users, which are children.

These building blocks environments support a fault-avoidant programming, which is especially important in the educational process. The physical representations of the building blocks suggest an exploration of the system. These systems are a great inspiration for educational methods and goals, but lack overview when the behavioral code grows.

GO WITH THE FLOW

Max MSP\(^3\) is a visual programming tool for the development of music, multimedia and interactive installations using execution-flow connections. It presents a balance between full access code and building blocks. It presents preset behaviors that can be important and reused throughout the tool. The flow controller interface provides a neat overview of the propagation through the code: top to bottom.

The flow controlled interface provides a great overview of the available behaviour. Code patches can be copy pasted within the global flow interface to hide their functionality. Max is a great tool hacking as it presents an overview of the system and enables a visual representation within the code. These properties enable users to understand where to change the functionality to adjust the code to their design brief.

Figure 2.30. Max MSP user interface

Max MSP\(^3\) - https://cycling74.com/products/max/.


2.4. Conclusion

Chopper focuses on the hacking autonomy phases of the design process. Demos within chopper should inherently suggest to explore and invite for iterations. Hacking tools should be made recognizable and allow for freely fault-avoidant exploration. Chopper should support the ability to learn from mistakes, test and iterate on existing code, verify the altered changes and work in incremental steps. The behaviour designs in Chopper should be quick to make and able to be provided timely. The effort of a hacking iteration should remain low, keeping each behaviour design disposable. The behaviors should inspire for new designs and invite for discussion, conversations and interactions.

Besides inspiration, motivation for designing a distributed network is key. Design guidelines and pitfalls should be apparent through the use of the tool. The opaqueness of the systems is important if the designer is to understand and apply global-to-local interactive qualities, key to distributed networks.

The presented examples should be inherently distributed to communicate the use of the global-to-local programming. These examples should include an interactive environment and an environmental interaction. Examples should include a gradient, a token and a custom distribution pattern. A vast amount of examples helps the designer understand the range of interactive qualities that can be achieved. Visualizing the example library helps to create a quick overview of this scope and inspire for new applications.

Each example should have at least some settings that are straightforward to modify. Modification tools should be recognizable and optionally hide troublesome details. Incrementally presenting more details as seen in NetLogo supports the incremental nature of the knowledge path of designers. Freely experimenting is made available through non-destructive interfaces such as buttons and sliders.

The distributed paradigm is a threshold concept, therefor users should be actively supported throughout the design process. Search windows, online communities and help buttons allow to do this. Similarly, recognizable syntax and icons should be used.
Design

List of requirements

Hypotheses II Design proposals

Final design

Scenarios of use
Technical details
This chapter introduces the design of Chopper: an educational software tool that introduces distributed networks to designers.

A list of requirements is extracted from the information as presented in chapter 2. Background. The design of the preliminary version of Chopper focuses on the hacking phase of the process. The list of requirements includes requirements to accommodate the most important phases in the design process: hacking, autonomy and nut-cracking, for future expansions.

Subsequently, this chapter presents a set of design proposals, or hypotheses, extracted from the list of requirements. These design proposals form the bases of the final design of Chopper.

Finally, this chapter presents the final design for the preliminary version of Chopper and illustrates the functionality through 3 storyboards: for the designer, the maker and the programmer.
3.1. List of requirements

A list of requirements states the important characteristics that the design of Chopper must meet in order to be successful (van Boeijen, 2013). It states the design objectives derived from the background research and is used to derive several design proposals. The list of requirements is divided into several categories to enhance readability.

### 3.1.1. GENERAL REQUIREMENTS

#### RECOGNITION RATHER THAN RECALL
1. Chopper presents a representation of the system in the real world by means of a simulation
2. Icons and syntax are re-cycled from familiar designer software tools where possible

#### VISIBILITY
1. Chopper’s system status is always visible, i.e. the status (on/off/paused) of the simulation and the distribution of new settings throughout the system
2. Chopper’s visible functionality is kept to a maximum of 20 actions in a screen
3. Chopper visualizes any alterations to provide immediate feedback
4. Chopper presents an incremental means of traversing through the level of details in alternative system views

#### REFERENCING
1. Chopper provides help & documentation: an overview of the example behaviours structure, application, functionality and potential alterations
2. Chopper presents a visual example behaviour library that provides a quick overview of the scope of the system

### 3.1.2. HACKING

1. A behaviour sketch – a global interaction through local rules – in Chopper is quick to produce
   a. Existing example behaviour sketches can be hacked to incorporate the design brief within 30 minutes.
2. A behaviour sketch takes little effort to produce
   a. A behaviour sketch can be disposed of within 10 minutes
3. Example behaviour sketches can be modified and iterated
   a. Example behaviour sketches contain 5 local settings that support alteration

#### TRANSPARENCY
1. Chopper visually connects the global behaviour to a local representation of the nodes in the network
2. Chopper supports the modification of the configuration of the autonomous nodes
   a. Chopper presents 2 alternative sensors and actuators
   b. Chopper presents 3 alternative aesthetics for the nodes
3. Chopper supports the distribution of alternative local settings throughout the network

#### EXPERIENCE
1. Users can freely experiment with example behaviors
   a. Chopper presents a minimum of 3 example behaviors
      i. An interactive environment and an environmental interaction

#### LEARNING
1. Chopper supports learning from mistakes
   a. Users can verify their new or altered code in the simulation
   b. Functionality can be added incrementally and reverted
2. Chopper presents recognizable modification tools
3. Chopper supports the comparison of alternative combinations of settings

### 3.1.3. AUTONOMY

#### CONFIGURATION
1. Chopper visualizes the autonomous progress of each node
2. Chopper visualizes the topology of the entire network
3. Chopper supports the reconfiguration of the topology
   a. Chopper supports the adding and removing of nodes and connections in the network
   b. Chopper supports the disabling and enabling of individual nodes in the network

#### PROGRAMMING
1. Chopper enables the user to build a behaviour sketch from scratch
   a. A user is able to product an autonomous sketch within an hour
2. Chopper supports the recycling of behaviour functionality amongst sketches
3. Chopper provides predefined functionality for the autonomous behaviour in visual building blocks
   a. Conditional triggers
   b. Output mapping
4. Chopper supports stepwise debugging of the visual programming code

#### DISTRIBUTION
1. Chopper visualizes the communication between nodes in the network
2. Chopper provides several preset distributed algorithms:
   a. A gradient algorithm
   b. A token algorithm
   c. A spatio-temporal distribution algorithm
3. Chopper supports the modification of existing distributed algorithms
4. Chopper supports the adding and removing of multiple distributed algorithms per behaviour
3.2. Hypotheses || Design proposals

The list of requirements forms the basis for the design proposals, or hypotheses, on how to enable designers in the understanding and creatively applying of interactive qualities of distributed networks. From this list, five design proposals have been extracted:

1. Providing a settings interface to change the behavioural rules of demo projects similar to that of NetLogo will allow designers to explore alternative interaction designs. This quantitative exploration will enable the designers to be more creative in their design process.

2. By creating opaqueness in the topology, designers will be able to understand and apply the interactive qualities of a distributed network.

3. Emphasizing the effect of communicating local autonomy enables the designer to understand and apply distribution patterns.

4. By providing alternative example distribution patterns, designers will be able to explore alternative interactive qualities. This will in turn enable designers to be more creative in their design process.

5. By traversing between a global view of the system and an alternative local view of the autonomous entities, designers will be able to understand and apply the interactive qualities of a distributed network.

These hypotheses are evaluated using the methodology as presented in chapter 4. Methodology and reflected in chapter 8. Conclusions.

The following sections illustrate each design proposal and their deployment within Chopper. Inherent to all design proposals is the division of functionality amongst alternative views. The level of details is divided amongst a global and local view of the system.
3. LOCAL AUTONOMY

The third design proposal emphasizes the local autonomy of the nodes in the network. Visualizing the autonomous progress and communication of each tile to collaborate on a global scale from a local perspective enables the designers to understand and apply the functionality of the global-to-local programming.

The focus on the local nodes enables the designer to understand and alter both the autonomous behaviour and the distribution of data in the network. A debug view that visualizes the communication and collaboration provides further insights in the distributed interactive qualities. It visualizes the amount of control global-to-local programming actually provides. The autonomous and distributed behaviour is presented in alternative pre-set behaviour and allows a maker and programmer to alter the global behaviour without having any experience in distributed networks.

As mentioned before, to inspire the designers many example behaviours should be presented. These behaviours should represent the entire scope of the available interactive qualities in a distributed network. All example behaviours should be inherently distributed and at least include the following properties:

- Interactive environment vs. environmental interactions
- Random vs systematic interactions
- Gradient vs. token vs. custom spatio-temporal distribution patterns.

Due to limited development time, the preliminary version of Chopper presents 2 example behaviors: “mowing the lawn” and “illuminate me”.

The first example project is an interactive environment representing an environment in which grass grows at a random speed and the user mows the lawn inspired by the Rabbits Grass Weeds example in NetLogo. The grass growth and the user mowing is distributed through the network with their own gradient properties – radius, minimum value and maximum value. Both gradients start off with a radius of 2 steps to enable the exploration of more interesting settings. Similar to real life, the grass is initially represented in shades of green.

The use of this example should enable the user to understand that the dependency of multiple gradients or distribution patterns can be visualized using the outputs. The most difficult aspect of this example is the random growth speed of every tile.

The second example is an environmental interaction. The walking direction of the user is detected by requesting the departure point from its neighbors. Subsequently, a path in this direction in illuminated in a spatio-temporal distributed spread. The distributed range of the data determines the time to fade back to the original color and the color setting changes the distributed color of the light. This allows to illuminate a path similar to seen in Figure 3.8, in an emergent fashion. This emergence will enable the users to understand the effect of small local interactions in the global representation of the system.

This example is not inherently distributed, but instead a response to the pilot study using Chopper. Participants in the pilot study that used the “mowing the lawn” example mentioned they had a lot of difficulty interpreting the randomness. Instead, they would have liked to see a more straightforward example. Hence, the “illuminate me” example.

4. EXAMPLE DISTRIBUTION PATTERNS

“MOWING THE LAWN”

The first example project is an interactive environment representing an environment in which grass grows at a random speed and the user mows the lawn inspired by the Rabbits Grass Weeds example in NetLogo. The grass growth and the user mowing is distributed through the network with their own gradient properties – radius, minimum value and maximum value. Both gradients start off with a radius of 2 steps to enable the exploration of more interesting settings. Similar to real life, the grass is initially represented in shades of green.

The use of this example should enable the user to understand that the dependency of multiple gradients or distribution patterns can be visualized using the outputs. The most difficult aspect of this example is the random growth speed of every tile.

“ILLUMINATE ME”

The second example is an environmental interaction. The walking direction of the user is detected by requesting the departure point from its neighbors. Subsequently, a path in this direction in illuminated in a spatio-temporal distributed spread. The distributed range of the data determines the time to fade back to the original color and the color setting changes the distributed color of the light. This allows to illuminate a path similar to seen in Figure 3.8, in an emergent fashion. This emergence will enable the users to understand the effect of small local interactions in the global representation of the system.

This example is not inherently distributed, but instead a response to the pilot study using Chopper. Participants in the pilot study that used the “mowing the lawn” example mentioned they had a lot of difficulty interpreting the randomness. Instead, they would have liked to see a more straightforward example. Hence, the “illuminate me” example.

Figure 3.4. A gradient distribution pattern

Figure 3.5. A token distribution pattern

Figure 3.6. A custom spatio-temporal distribution pattern

Figure 3.7. The Rabbits Grass Weeds example in NetLogo (Wilensky, 2001)

Figure 3.8. Van Gogh interactive Starry Night bikepath (Roosegaarde 2014)
5. TRAVERSING

Inspired by the alternative view modes in NetLogo, Chopper provides different views in the system to hide/show functionality of the system and caters to the different goals within the design process. In a similar fashion, these views integrate the local nodes into a global behaviour, creating an opaqueness between them. The last design proposal enables the traversing amongst the different views, inspired by the global-to-local transparency.

GLOBAL VIEW

Presents the global experiential simulation of the system. It supports visualization of the topology and the enabling and disabling of the network of individual nodes.

GLOBAL-TO-LOCAL VIEW

Presents the functionality to alter local rules and distributed them globally throughout the network.

LOCAL VIEW

Presents the local configuration a selected node and its neighboring nodes in the network. This debug view presents provides insights in the distribution of data in the network and the autonomous setup of the nodes.

LOCAL-TO-LOCAL VIEW

Presents the local rules of communication and collaboration between the nodes in the network. It provides full access to the distributed algorithms and supports the alterations of the patterns. The local-to-local view is managed directly by the programmer and is of essence in the nut-cracking phase of the design process. The local-to-local view is not developed in the preliminary version of the tool, but recommended for future works.
3.3. Final design

3.3.1. GLOBAL-TO-LOCAL VIEW

The global view of the simulation uses a Z-pattern as seen in Figure 3.10 to structure the most important functionality. The Z-pattern states that a user starts by looking in the top-left corner, follows a Z-shape and finally focuses on the right-bottom corner of the screen. Basic navigational tools are positioned in the top-left corner to guide the user into an initial navigation through the system. The top-right corner presents drawing functionality of the on/off states and is encouraged for use. The bottom-left corner is intentionally left blank to support the processing of the simulation. Finally, the local settings interface is presented in the bottom-right corner to encourage the exploration of different settings.

After exploring the Z-pattern, the user is encouraged to discover new functionality in the top-middle of the header. The slider represents the level of detail and interaction with the slider allows the user to traverse through these levels.

The simulation of the behaviour is centralized and most important in the global view. It allows the user to visually connect the dots in the many dotted notion of distributed networks. Players can be selected to move, be removed or followed in a third-person perspective for a direct experience of the interactions.

The slider allows traversing through the different views of the system. The slider is only visible when a tile is selected – visible in magenta. The view focuses on the selected tile when the slider is moved. Right underneath this abstract form of traversing, several preset perspectives are provided to change the perspective of the simulation in P1, P2, P3 and P4.

The global settings provide a canvas to globally change variables within the separate entities in the system, such as the on/off variable. The user can toggle this state for each individual tile in the global construct. This tile will turn black and is unresponsive to the players and the neighboring tiles in the system.

Local settings are static settings similar for each individual node in the system that determine the global behaviour. The settings are visualized in the simulation and by using icons. The user can use the recognizable medication tools to explore new behaviour. The changes become apparent when the settings are distributed throughout the network.
3.3.2. EXAMPLE BEHAVIOURS

Figure 3.11 illustrates the behaviour of example 1: "mowing the lawn". The user walks around in the interactive environment using a player. The player mows the lawn with a small gradient surrounding him. The grass grows back at a random speed and distributes in a separate gradient.

Figure 3.12 illustrates the behaviour of example 2: "illuminate me". The user walks around in the simulation and initiates an environmental interaction as seen as the player interacts with the tiles. Dependent on the direction of the player, a path is illuminated to guide the player.
3.3.3. LOCAL VIEW

The local view of the system represents the autonomous nodes within the network. It visualizes the communication and collaboration of the autonomous nodes through a distribution canvas. This canvas provides debug tools to step through the distribution of the data and explore the effect of local interactions in the system. Additionally, the local view presents an abstracted version of the tool to help enable the programmer designers in understanding the functionality.

The distribution canvas represents the data state for the selected tile and its neighboring tiles. Important distributed variables are shown; the grass gradient and the user gradient in this case. It provides a canvas for testing the distribution of these variables. The user can initiate the grass to grow and step through the code to visually see the distribution of data.

The debug tools enable the user to show and hide details within the local view. It allows stepping through the code and the undoing of changes.

The code is presented in abstract visual code. The progress bar left of the code shows the current state within the code for the selected tile. It represents important variables that would optimally be open for modification in a finalized version of Chopper. It presents an explanation of the code when the user hovers over the code line.

Figure 3.13  The local view in the preliminary version of Chopper
3.3.4. SCENARIOS OF USE

The following sections illustrates the functionality of the preliminary version of Chopper, as used in the pilot explorative comparative study, by means of a story board. A walk through of typical use cases for the most significant persona in the research is presented: the designer, the maker and the programmer.

The story boards portray the different intentions of use for each persona through the actions they perform in the tool. The designer focuses on experiencing the interactions with the distributed network through the addition and exploration with players, different perspectives, playful state drawing and the distribution of a new color palette. The maker focuses on the structural properties of the network through visualization of the connections, turning nodes on and off, distributing altered autonomous node settings and exploring the distribution of data. Finally, the programmer explores the gradient settings through visualization and trial and error and tries to understand both the code and the distribution of data in the network.

THE DESIGNER STORYBOARD

Figure 3.14. Open an example
Figure 3.15. Explore the interaction
Figure 3.16. Add players
Figure 3.17. Drag players

Figure 3.18. Drop players
Figure 3.19. Choose a perspective
Figure 3.20. Get inspired
Figure 3.21. Create use cases such as grouping together

Figure 3.22. Access a player
Figure 3.23. Follow the interaction
Figure 3.24. Explore drawing tool
Figure 3.25. Draw for fun

Figure 3.26. Select a tile
Figure 3.27. Change the color settings
Figure 3.28. Distributed the settings
Figure 3.29. Experience the changes
THE MAKER STORYBOARD

Figure 3.30. Open an example
Figure 3.31. Visualize the topology
Figure 3.32. Understand the topology
Figure 3.33. Explore deployment

Figure 3.34. By turning the system off
Figure 3.35. Drawing with exploring intensities
Figure 3.36. Experience the effect of turning tiles off
Figure 3.37. Change autonomous behavior settings

Figure 3.38. Visually see the proximity threshold
Figure 3.39. Distributed new settings
Figure 3.40. Experience the change in behavior
Figure 3.41. Traverse to the local view

Figure 3.42. Open the debug panels
Figure 3.43. Experience the global-to-local transparency
Figure 3.44. Initiate the distribution of grass
Figure 3.45. Experience the distribution

THE PROGRAMMER STORYBOARD

Figure 3.46. Open an example
Figure 3.47. Explore the gradient settings
Figure 3.48. Visually see the gradient settings
Figure 3.49. Alter the radius of the gradient

Figure 3.50. Visually see the radius setting change
Figure 3.51. Distribute the settings through the network
Figure 3.52. Experience the change in settings
Figure 3.53. Traverse to the local view

Figure 3.54. Try to understand the code through explanation
Figure 3.55. Open up the debug view
Figure 3.56. Enable the detailed view
Figure 3.57. Initiate the growing of grass

Figure 3.58. Experience the autonomy
Figure 3.59. Step through the code
Figure 3.60. Experience the distribution
Figure 3.61. Experience output respond
3.3.5. TECHNICAL DETAILS

Chopper is developed in Unity\(^1\) 4.6.2f1. Build for PC, Mac & Linux Standalone on a Windows target platform x86 in the C# programming language.

The individual nodes in the interactive square are developed as prefabs – prefabricated models with aesthetics, functional code and animations. Especially for the homogeneous nature of the distributed networks, this deployment works well.

Each node is equipped with the following C# classes:

1. A neighbor script: that connects the tile to neighboring tiles within a predefined range and stores them in a list.
2. A global behaviour script: the behaviour of the tiles that causes the global transparency. This script calls all other scripts.
   a. A proximity script: calculates the lowest proximity measure amongst all players in the system.
   b. A first gradient: that calculates the grass gradient based on its own and neighboring states.
   c. A second gradient: that calculates the user gradient.
   d. An output color script: that translates the grass variable and renders an appropriate color.
   e. A distribution script: that propagates the local settings by storing versions numbers amongst tiles.
3. A mouse control script: that handles the navigation with the mouse in the simulation.
4. A local behaviour script: the behaviour when the tiles are in focus in the local view for debugging.
   a. A first local gradient script: that handles the local representation of the grass gradient.
   b. A second local gradient script: that handles the local representation of the user gradient.
5. A configuration script: that keeps track of the current position of the tile according to its x and y.

Methodology

Significance

Instructions

Research setup
- Participants
- Test cases
- Design brief
- Units of analysis
- Procedure

Criteria
- Understanding
- Creativity
- Application

Expectations
- Quantitative
- Qualitative
This chapter presents a generalized methodology — a construct of methods and criteria — for evaluating tools and methods that educate in the context of design. The methodology is illustrated with the explorative comparative study that is set up to evaluate the effect of Chopper in introducing distributed networks to designers.
The methodology developed to evaluate Chopper uses a multi-disciplinary approach to evaluate the effect of specific educational tools and methods within the context of design. The methodology is recommended and necessary for evaluating the effect of educational tools and methods within threshold concepts and troublesome knowledge in the context of design. However, the methodology could be used for any educational goals within the context of design: to evaluate specific aspects within tools and methods, to evaluate the overall effect of a tool or methods or to evaluate and compare several tools and methods to design an educational curriculum.

The quantitative criteria that support the evaluation methodology are derived from the initial assumption in this thesis:

If designers are able to explore and understand interaction designs for distributed networks, they will be able to produce creative applied interactive qualities for distributed networks in turn.

The methodology combines the subject specific measure of the participants’ understanding with the design context related measures. The design context relates to the design driven innovation of a technology push, both in creativity and the application of the subject matter at hand. Each criterion is further explained in the criteria section.

To accommodate reliable results, the methodology presents instructions for use and a recommended research setup inherent to the design process. Each step in the methodology is illustrated with the explorative comparative pilot study that is setup to evaluate the design proposals within Chopper.

**EXAMPLE RESULTS**

The methodology provides both quantitative and qualitative result. Quantitative results are summarized in a graph for each criterion: understanding, creativity and application. The methodology compares the effect of methods and techniques deployed in several test cases. A successful test case example results is presented in Figure 4.2 where both the understanding, the creativity and the application of the subject matter is improved. For more expected outcomes of the research, please read the expectations section.
The following sections present an overview of the methodology. Each section is illustrated with the explorative comparative study used to evaluate Chopper. This explorative study is limited to making suggestions as the size of the research is too small. However, each section will provide enough information to setup both a pilot and a large scale study to evaluate an educational tool or method in the context of design.

The methodology describes the following steps:
1. Determine the target audience
2. Determine the educational goals
   a. Define an indication for when the goals are reached
3. Design or choose an educational tool or method to accommodate these goals keeping the previous in mind
4. Extract the independent methods and techniques that should be evaluated
   a. Formulate these into several comparative test cases
5. Structure a design brief with limited variability
6. Design a design template inherent to the subject matter
7. Setup an exam that evaluates the understanding of the subject matter
8. Categorize a representation of the total participant demographic for each test case
9. Conduct the evaluation research
10. Evaluate the exams and evaluate the designs using experts
11. Evaluate the methods and techniques and provide recommendations for future work

This chapter presents the criteria that are significant within the evaluation methodology to evaluate a subject matter in the context of design. It then presents and illustrates the required research setup to extract the quantitative and qualitative results. For recommendations on applying the methodology it is recommended to read Chapter 7. Recommendations.
4.3. Criteria

This section describes the methodology specific criteria to evaluate the effect of an educational software tool within a specific subject matter in design.

It presents the understanding of a specific subject matter within the context of design. It presents a redefinition of creativity as the second criterion and finally presents the application of the specific subject matter within the context of design.

4.3.1. UNDERSTANDING

The understanding of any subject matter is ambiguous. To assess understanding, numerous and diverse forms of assessment are required (Wiggins, et al., 2005). Understanding a subject matter is different from merely possessing the knowledge and is essential in order to master threshold concepts. Figure 4.3 provides an overview of these differences.

The difficulty in understanding threshold concepts leaves students in a state of threshold. This state of threshold leaves them to either approximate or replicate the concept through examples in the subject. These approximations lack correct use and the replications lack authenticity and creativity (Meyer, et al., 2003). Therefore, the first criterion in the evaluation methodology evaluates the difficulty in understanding.

The six facets of understanding (Wiggins, et al., 2005) are used to evaluate the overall understanding of a designer within the subject matter, or threshold concept. The facets are evaluated through a written exam. Each facet is represented as an open question and evaluated using an answer sheet prepared prior to the research. Finally, each participant will receive a grade ranging from 1 – 10 to represent their level of understanding.

1. The grade from 0 – 10 from the subject matter related exam

Each question should be essential and push to the essence of the subject matter. Generate each question with great care, keeping learning goals in mind at all times. The following criteria are provided as a guideline for developing essential questions (Wiggins, et al., 2005):

- Cause real and relevant inquiry in the big ideas and core content
- Provoke deep thought, lively discussion as well as more questions
- Require participants to consider alternatives, support their ideas and justify their answers
- Stimulate rethinking of big ideas, assumptions and prior learning and experiences
- Spark meaningful connections with prior learning and experiences
- Naturally create opportunities for transferring to other contexts

<table>
<thead>
<tr>
<th>KNOWLEDGE</th>
<th>UNDERSTANDING</th>
</tr>
</thead>
<tbody>
<tr>
<td>The facts</td>
<td>Meaning of facts</td>
</tr>
<tr>
<td>A body of coherent facts</td>
<td>Theory proving coherence</td>
</tr>
<tr>
<td>Verifiable claims</td>
<td>Theories that are in process</td>
</tr>
<tr>
<td>Right or wrong</td>
<td>A matter of degree of right or wrong</td>
</tr>
<tr>
<td>Knowing something to be true</td>
<td>Understanding why it is true</td>
</tr>
<tr>
<td>Respond on cue with that you know</td>
<td>Judgment when and when not to use knowledge</td>
</tr>
</tbody>
</table>

Figure 4.3. Knowledge vs understanding (Wiggins, et al. 2005)
THE 6 FACETS OF UNDERSTANDING

This section describes and illustrates the six facets of understanding (Wiggins et al., 2005)

1. **Explanation**: can explain via generalizations or principles, make insightful connections and provide illuminating examples or illustrations.
2. **Interpretation**: can interpret and tell meaningful accessible stories through images, anecdotes, analogies and models.
3. **Application**: can apply what is known in diverse and real contexts – “do” the subject.
4. **Perspective**: have perspective to see the big picture through critical eyes and ears.
5. **Empathy**: can empathize and find value in what others might not perceive.
6. **Self-knowledge**: have self-knowledge and are aware of what they do not understand and are able to reflect on the meaning of learning and experience.

The illustrates example questions for each facet in the explorative pilot study evaluating Chopper are presented in the teal illustrations.

WRITTEN EXAM FOR DISTRIBUTED NETWORKS

A quantification of each participant’s understanding of the interactive qualities of a distributed networks is made through the following exam questions. Each exam question is essential and matches a facet of understanding:

1. Explain in your own words as extensively as possible what a distributed network is (think of its components, its functionality and its cause).
2. Birds show an overall emergent behaviour by flying in flocks. By generalizing the flock as a distributed network, can you explain how this works? Add sketches, models and diagrams to help your explanation.
3. Name an existing system that would improve from switching to a distributed network. How does this system work now, how would it improve and how would the final system work?
4. Can you explain in general terms when it would be appropriate to design an interaction using a distributed network as opposed to other solutions?
5. Name at least 1 strength and 1 weakness of a distributed network. Explain why these are strengths and weaknesses.
6. You’ve developed a number of interaction designs. Please score your interaction designs on the application of interactive qualities for a distributed network on a Likert Scale (as presented in section 4.3.3). Explain why.

QUANTIFICATION

Each facet of understanding is equally important, hence each facet of understanding should be represented equally in the exam. A large number of questions will result in smaller variability due to question formulations, but requires more time to complete the exam.

An answer sheet is prepared prior to the research indicating the core concept required to answer each question. Based on the answers, a score is provided for each question ranging from 0 – 4 points. The final grade is calculated by dividing the total score for each participant by the total available score.
4.3.2. CREATIVITY

To assess a subjective criteria such as creativity, numerous and different forms of assessment are required. Creativity has many different definitions. This criteria is redefined to include the design-driven innovation for technology epiphanies and to accommodate creativity in the design process using the following metrics:

1. The increase in interaction designs
2. The increase in innovative interaction designs
3. The increase in most innovative design
4. The increase in authentic designs
5. The increase in different designs

An increase is measured to accommodate for any inter-participant variability that may arise due to skills and experience opposed to differences due to the educational methods. The first metric is fairly straightforward: a quantified count of the design templates that use all designated areas. Metrics 2-3 need an underlying score for innovativeness and metrics 4-5 for similarities.

Variability may occur due to their interpretations of the evaluation task. Similarly, experts received the design assignment inherent to the produced designs to evaluate the innovativeness of the solutions in the context of the problem. Creativity is a subjective measure and priming the experts for creativity is therefore unwanted. Each designer receives the total amount of designs in a random order to limit any order bias and is asked to score the designs on the creativity Likert scale in Figure 4.4.

The number of total interaction designs is a total count of the designs produced by each participant. The number of innovative designs is quantified by the number of designs that are on the agree part or strongly agree part of the Likert scale. The most innovative design is the design that is ranked highest on the creativity Likert scale.

The authenticity of each design is measured through the sorting of the designs based on their similarities. The nature of an authentic design is the low occurrence of similar designs. The authenticity metric requires a sorting of the designs based on similarities in their interactions. Designs that are paired less often are thus more authentic. The authenticity of the designs is not a representative measure alone as a singular represented design might also be a bad design. After scoring each design on the Likert scale, design experts are asked to sort each subsection of the scale on the similarities in the interactions in as many groups as possible.

Figure 4.5 presents the white participant that has developed a total of nine designs of which three are innovative and the most innovative scoring design is a strongly agree. The white participant has developed a single authentic design.

The green participant has developed eight designs of which three are innovative and the highest scoring design is also a strongly agree. The green participant has produced two authentic designs.

While variability is hard to limit within a subjective measure such as innovativeness, it must be taken into consideration while extracting these measures. Results of the evaluation are presented in chapter 5. Results.

<table>
<thead>
<tr>
<th>Strongly disagree</th>
<th>Dissagree</th>
<th>Neither agree nor disagree</th>
<th>Agree</th>
<th>Strongly agree</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This interaction design is an innovative solution to the design brief:

Figure 4.4 The Likert creativity scale

2. IDEO: http://www.ideo.com/about/.
QUANTIFIED CREATIVITY

The design experts are selected on their experience in evaluating designs in education. If and when their opinion differs, all experts may be right in different aspects within design. While the evaluation guidelines limits the variability induced due to interpretation of the design brief, design experts can have different perspective on the term innovativeness. It can be argued that as long as a single design experts evaluates a design as innovative, it is innovative in at least some aspect of the design. When a small group of experts is used for evaluation, scores can have much variability amongst them. Using a mean or median representation results in an average score for every design, indicating the need for a mode representation of creativity. However, when many design experts are available, the overall trend in a mean will be a better indication.

Creativity is a finite source of variation. Initially, each concept generation phase produces slightly similarly creative results. Previous concepts may inspire subsequent phases and ideas may be repeated. However, within a time limited structure, creativity is expected to decrease over a period of time without new inspirational sources.

A slight decrease in the number of designs indicates fatigue in creativity due to the long duration of the research. Subsequently, a constant amount of designs during the concept generation phases indicates the lack of fatigue, a new found inspiration and is a positive outcome of the research.

![Expected trend in creativity](image-url)

Figure 4.6. Expected trend in creativity measures over time
4.3.3. APPLICATION

To assess the application of the subject matter, subject specific evaluations are required. The following metrics are designed to evaluate the application criteria:

1. The increase in the number of applied designs
2. The increase in the average application
3. The increase in the highest scoring design

Again, an increase in the metrics is evaluated between phases to limit any variability induced due to skills and experience of the participants.

The number of applied designs represents the ability of the participants to apply the interactive qualities. The average application of interactive qualities represents the mindset of the participant and differentiates between a focus on the application or an accidentally single applied design. However, similar to the creativity, merely a single applied design is required to produce a successful final product within the context of the subject matter and is taken into consideration as the third metric.

Specifically in the context of education in design, it is important to apply the learning goals in concept designs. The application of the subject matter is dependent on the context and requires experts in the subject field to evaluate the application of the subject in the designs. Again, these experts are experienced in evaluating applications specific for the subject.

This interaction design applies the educational goals of a subject matter, for example:

This interaction design applies interactive qualities of a distributed network:

<table>
<thead>
<tr>
<th>Strongly disagree</th>
<th>Disagree</th>
<th>Neither agree nor disagree</th>
<th>Agree</th>
<th>Strongly agree</th>
</tr>
</thead>
</table>

Figure 4.7. The Likert application scale

The application is evaluated by subject specific experts on a Likert application scale as seen in Figure 4.7. Experts receive guidelines for evaluating the designs and are primed to ensure limited variability within the group of experts for this objective measure. The total amount of designs in the research is presented to the experts in a random order to limit any order bias and is asked to evaluate the Likert application scale.

The increase in the number of applied designs is evaluated using a count of the designs in the agree part or the strongly agree part of the scale. The average application score is the average score of the application for all designs. Finally, the highest scoring design is the design that obviously scores highest on the scale.

Figure 4.8 illustrates an example rating for the application of the interactive qualities of a distributed network in the designs produced by this participant. The participants have produced five appropriate designs, two inappropriate designs, and two designs that are neither inappropriate nor appropriate. The average score is around just above the neither agree nor disagree category and the highest scoring design is a strongly agree.
The explorative comparative study to evaluate the effect of the design proposals used three experts in the field of computer science to evaluate the application of the interactive qualities of a distributed network.

The first two distributed network experts are the technical assistants from the StudioLab\(^1\) at the faculty of Industrial Design Engineering at the Delft University of Technology. Both experts have a background in Computer Science and are familiar with distributed networks.

1 StudioLab - Delft University of Technology. - http://studiolab.ide.tudelft.nl/.

Distributed network expert number three has extensive knowledge and experience in distributed networks, also in the context of design.

Each expert received a set of evaluation guidelines and the redefinition of distributed networks as presented in Chapter 2. Background. Additionally, experts are primed using the example behaviour sketches as seen in Chopper, represented in the design templates. Example one is presented in Figure 4.9 and example two in Figure 4.10.

Each expert is asked to evaluate both examples prior to the evaluation of all designs. Example one is an interactive environment and uses both a random factor, direct and indirect interactions with the users and 2 gradient patterns. It portrays the interactive qualities of a distributed network and should be categorized in the strongly agree. Example 2 is an environmental interaction that only responds to the user directly. It does propagate a message throughout the system, but doesn’t invite for interaction of multiple users and should be categorized in the agree part of the scale. Experts are talked through the evaluation guidelines until both the researchers and the experts agree.

The inter-rater reliability is defined by the agreement on the “official” rating of the design: experts should all agree or disagree that the design applies interactive qualities of a distributed network. Hence, experts should rate on similar sides of the Likert scale. This means that ratings are reliable if ratings are no more than one section on the scale apart. When ratings show larger variance, the evaluation guidelines for the raters should be adjusted appropriately.

If the number of experts is large enough and the variability between the ratings is limited, the median of the ratings is the optimal representation. The raters should agree on the “official” rating – either all agree or all disagree, so the sides of the Likert scale. Each rater should agree on the visibility or lack of interactive qualities of a distributed networks. Inherent to this agreement, the mode of all ratings is used to represent the score of each design. If the amount of experts is limited, the median is the next best measure.
First and foremost, the educational content in question is defined. Whether it are the methods and techniques deployed in a tool, the overall effect of the tool or several tools that need comparison, the independent elements for evaluation should be extracted and defined prior to the research. A separate test case represents the current available educational content in the subject matter. This test case enables the comparison between the effects of the educational tool or method and the currently available educational matter.

A realistic design brief is required to invoke a real design process in the subject matter involving the educational tool or method. The design brief limits any variability that may be induced due to interpretations. It specifies the use of the subject matter in the designs to reduce variability, but should still contain enough room to support creativity.

The demographic of the participants in the research is defined prior to the research. Each test case contains a representation of the total demographic to limit any inter-participant induced variability within test cases. Results are evaluated on the concept designs resulting from the design process. The understanding is measured through a written exam. Qualitative results are extracted using out-loud thinking, observations and interviews throughout the research.

This section describes a walk-through of an educational tool evaluation using the methodology. It illustrates each example with the pilot evaluation of Chopper.
4.4.1. PARTICIPANTS

The designer demographic contains designers, maker, programmers, managers and directors. Within each persona, variability is still apparent. Skills and previous experiences are determined within the demographic. Develop an explicit questionnaire to evaluate these experiences and skills. Categorize a representation of the demographic in persona, experiences and skills to each test case. If the research size is large enough, each test case can contain smaller groups for each persona to evaluate the inter-persona differences.

The pilot test uses 8 participants with a diverse demographic range. Prior to the research, an Online questionnaire is filled in to categorize participants in two test cases. Questions include age, gender and educational levels, but also the years of programming experience, their primary and secondary role in a design team and their experience in prototyping.

To ensure a normalized division of designer persona across all test cases, participants are categorized beforehand based on their answers on the questionnaire. The answers of the questions in the Google form are shuffled to prevent order bias.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>DEMOGRAPHIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age</td>
<td>22 - 30</td>
</tr>
<tr>
<td>Gender</td>
<td>Male / Female</td>
</tr>
<tr>
<td>Level of education</td>
<td>BSc / MSc</td>
</tr>
<tr>
<td>Degree</td>
<td>Industrial design engineering / other</td>
</tr>
<tr>
<td>Roles in teamwork</td>
<td>Designer / Maker / Programmer / Manager / Director</td>
</tr>
<tr>
<td>Years of programming exp.</td>
<td>0 - 5</td>
</tr>
<tr>
<td>Experience in tools</td>
<td>Arduino / Max MSP / Grasshopper / Scratch / Mindstorms / 3D modelling software / Eclipse or another development IDE</td>
</tr>
<tr>
<td>Experience in tools</td>
<td>1. A single microcontroller (Arduino, Phidgets, etc.) using sensors and actuators</td>
</tr>
<tr>
<td></td>
<td>2. Two connected microcontrollers using sensors and actuators</td>
</tr>
<tr>
<td></td>
<td>3. A central microcontroller controlling &gt; one microcontroller using sensors and actuators</td>
</tr>
<tr>
<td></td>
<td>4. More than two microcontrollers using sensors and actuators and the same software controller in a distributed fashion</td>
</tr>
</tbody>
</table>

Figure 4.11 The demographic of the participants in the explorative comparative pilot study

<table>
<thead>
<tr>
<th>PARTICIPANT NUMBER</th>
<th>3</th>
<th>7</th>
<th>10</th>
<th>5</th>
<th>6</th>
<th>8</th>
<th>9</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group number</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test case</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Age</td>
<td>25</td>
<td>25</td>
<td>20</td>
<td>22</td>
<td>30</td>
<td>24</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>Gender</td>
<td>F</td>
<td>M</td>
<td>F</td>
<td>M</td>
<td>F</td>
<td>F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Level of education</td>
<td>MSc.</td>
<td>MSc.</td>
<td>BSc.</td>
<td>BSc.</td>
<td>MSc.</td>
<td>MSc.</td>
<td>MSc.</td>
<td></td>
</tr>
<tr>
<td>Degree</td>
<td>IDE</td>
<td>IDE</td>
<td>IDE</td>
<td>IDE</td>
<td>IDE</td>
<td>other IDE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Role 2 in design teams</td>
<td>Mak.</td>
<td>Prog.</td>
<td>Man.</td>
<td>Prog.</td>
<td>Dir.</td>
<td>Prog.</td>
<td>Mak.</td>
<td>Prog.</td>
</tr>
<tr>
<td>Years of programming exp.</td>
<td>4</td>
<td>5</td>
<td>0.5</td>
<td>1</td>
<td>0</td>
<td>0.5</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>Experience in programming</td>
<td>1,2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1,2,3</td>
</tr>
</tbody>
</table>

Figure 4.12 The participants in the explorative comparative pilot study
4.4.2. TEST CASES

Each test case represents an independent element that is compared in the study. This could be a method or technique deployed in the tool, the full tool, or even several different tools or methods. Care must be taken when formulating these test cases to ensure the proper comparison is made.

A separate test case should be defined prior to the research to illustrate a base line for comparison. A base line is a current context of (the lack of) educational tools and methods which the educational tool tries to improve, or a minimum functionality to which other elements in the tool are compared to.

The methodology can be applied in early phases of the design process of the educational tool or method. This means that the functional scope of the preliminary version can be limited. Determine the functional scope of the method and tool that is required in order to test the test cases prior to the research.

There is currently no tool available that introduces distributed networks to designers. To evaluate the effect of the methods and techniques deployed in Chopper, a bare minimum representation of the functionality within Chopper is extracted. This bare minimum is used as a base line for comparison of the explorative comparative study. The design proposals deployed in the preliminary version of Chopper are extracted and filtered into the following test cases:

**GLOBAL SIMULATION**

The global view contains a real-time simulation of the available example behaviours. It provides the functionality to show/hide connections in the topology and to turn each individual node on/off toggle. This functionality displays some interactive qualities inherent to the global-to-local programming and should induce some initial understanding.

**LOCAL SETTINGS**

The same global view now offers an interface that enables the modification of settings in the local code. The visual representation of these settings enables the user to explore more interaction designs, stimulating creativity. The distribution of these settings is visualized in the simulation and enables the user to experience the global-to-local programming. The direct interaction with the local settings should not only increase their understanding, but enable them to apply the interactive qualities in their designs. Quantitative changes in the criteria are verified using an interview to evaluate whether the designer experienced changes, was able to explore the interaction designs and experience any changes within the criteria.

**LOCAL VIEW**

The local view enables users to traverse between global and local representations of the behaviour, increase the understanding of the global-to-local transparency. A better understanding of the distribution of the data in the system hypothetically leads to proper application of distribution patterns in the produced interaction designs. To verify the effect of the added functionality a small exam is performed right after phase 3 to explicitly test how well the participants understand the interactive qualities of a distributed network.
4.4.3. SCOPE

Not all aspects of the tool need to be functional in order to test the effect of the test cases. Determine the essential methods and techniques within each test case and evaluate what has to be functional in order to evaluate its effect.

Merely suggesting the possibility of changing a value could be enough in this case. The following categories are not implemented in the preliminary tool:

- Adding and removing nodes
- Alteration of operational statements
- Alteration of available IO
- Alteration of the distribution flow
- Alteration of the aesthetic of a node

Reference sheets are provided to ensure there is minimum variability within the functionality that is found in each version of the tool.

4.4.4. DESIGN BRIEF

A design brief entails the assignment for the designer and specifies the project’s requirements and wishes. A design brief is required to emulate a realistic context for the design process. It presents enough information about the design context, goals, limitations and target users and inspires the designer.

Inherent to the comparative nature of the study, any variability induced by factors other than the compared methods and tools should be limited. The design brief should remain open enough for the designer to produce creative designs, but should leave limited room for variable interpretations and subject matter unrelated issues.

The explorative comparative pilot study evaluates the educational effect of Chopper in the context of distributed networks. The educational goals are limited to applying the interactive qualities of a distributed network in the produced designs. To support this focus, any variation in aesthetics, positioning, design goals, environmental context and the target users is limited and provided in the brief as presented in Appendix 9.1 Design brief. The design context used to evaluate Chopper is the Catharinaplein in Eindhoven. Eindhoven has advertised itself as a testing group for intelligent lighting and has introduced a road-map to have integrated intelligent lighting throughout the city that contributes to the quality of life.\(^1\)

The design brief introduces a given distributed grid of interactive floor tiles. Each participant is asked to produce as many creative designs fit for the provided network. Specific design goals are provided: create an invitation for social interaction between the tourists and citizens and simulate an invisible infrastructure that invites users to explore Eindhoven beyond the square.


Figure 4.14 The reference sheet for the global-to-local view illustrating the functional scope of Chopper.

Figure 4.15 Google world view from the Catharinaplein in Eindhoven.
4.4.5. UNITS OF ANALYSIS

This section introduces the units of analysis applied in the methodology. These units are both quantitative and qualitative and diverse to represent a complete picture of the effect of the educational method.

OBSERVATIONS

Participants are asked to think out loud during the entire research. This enables the researcher to note and observe any issues or insights. All questions that are asked during the research are documented for further evaluation. Questions that are specifically to subject matter indicate a certain level of understanding the participant has (Wiggins, et al., 2005). If the questions are superficial, aesthetic questions, it indicates a lack of understanding.

Besides the variability induced by the different persona, the interaction with the tool can induce a range of variability in the design results. Concurrently, users should be constantly reminded to illuminate their motivations behind the actions during the research.

INTERVIEWS

Interviews are used to evaluate any specific behaviour that may have come up during the research. These observations are written down and evaluated after each concept generation phase. A concluding interview is used to evaluate the specific educational goals that should be achieved during the research explicitly.

DESIGNS

During the design process, participants are asked to produce as many designs in accordance to the design brief. These designs are evaluated by experts on their creativity and the application of the subject matter. Any inter-participant related differences unrelated to the test cases should be limited. To do so, the designs have to be structured in a design template.

Inter-participant differences can induce variability in the evaluation of these designs due to:

- Drawing skills
- Representation of the context
- Level of detail
- Applying the subject matter
- Answering the design brief

To limit this variability, a design template is used that designates space for detailing and reduces the effect of drawing skills. The template presents a designated space to answer the design brief, present the global representation of the design and a representation of the specific deployment of subject matter within the design.

The design template used in the pilot study displays 3 designated spaces: a global representation of the users and the system, a functional specification of the network and a space to answer the design brief. User stickers are provided to eliminate the need for drawing users within the system.
### 4.4.6 Procedure

The research procedure is a strictly time-limited procedure of conducting the research. It should limit any variability due to time differences. Each participant follows the exact same procedure, except for the assigned test case. Participants in the base line procedure use the same functionality of the educational tool for the entirety of the research. Participants in the procedure that evaluates the design proposals get added functionality inherent to the test cases in each phase of the research. The comparison between these test cases evaluates whether the increase is indeed due to added functionality or due to the use of the tool for an extended amount of time.

By dividing the procedure into several test phases, a differentiation can be made between initial use of the tool and methods and the participant dependent skills and expertise that should be visible in the first phase of the procedure.

<table>
<thead>
<tr>
<th>Phase</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phase 1</td>
<td>20 min brief, 20 min use, 10 min concept generation, 20 min interview.</td>
</tr>
<tr>
<td>Phase 2</td>
<td>20 min brief, 20 min use, 10 min concept generation, 20 min interview.</td>
</tr>
<tr>
<td>Phase 3</td>
<td>20 min brief, 20 min use, 10 min concept generation, 20 min interview.</td>
</tr>
</tbody>
</table>

**Figure 4.18** The bare minimum procedure

**Figure 4.19** The design proposals procedure

---

**TAKE NOTICE:**

The explorative comparative pilot study was conducted to evaluate Chopper as well as evaluate the overall methodology. The addition of design proposal 2 in phase 3 as seen in test case 2 however will not provide reliable results. The added functionality is added on top of the already added functionality. The results may be due to the extra added functionality or due the use of the functionality added in phase 2 for a longer amount of time. For recommendations of improvement please read Chapter 7: Recommendations.
4.5. Expectations

This section presents what can be expected as an outcome of the evaluation. Reading instructions for quantitative results are provided as well as qualitative explanations. The effect of the educational tool may take multiple directions. The understanding of a threshold concept may inspire and motivate the designers to start applying the concept. It may also destroy their initial ideas and perception of the concept and de-motivate them entirely.

The truth or insight may be a pleasant awakening or rob one of an illusion; the understanding itself is morally neutral. The quicksilver flash of insight may make one rich or poor in an instant (Palmer, 2001)

Effectively evaluating the introduction of distributed networks might still not mean the introduction of the concept is wanted. Qualitative results should accommodate for these insights.

4.5.1. QUANTITATIVE

The nature of an educational goal in the context of threshold concepts is an increase in the understanding of the subject matter. The nature of an educational goal in the context of design is an increase in the understanding of the subject matter followed by the ability to apply the subject matter in design. Consequently, the best case results present an increase in both understanding and application compared to the base line version of the tool. Ultimately, the increase of creativity implies that participants do not focus solely on the tool and the subject matter, but also focus on integrating the subjects matter in their concepts.

On the contrary, the worst case results would be any case in which the understanding of the subject matter is less than the understanding of the participants using the base line version of the tool. If the understanding is similar or less, any change in the application cannot be explained without qualitative results. In any case where the understanding decreases and the creativity increases, it implies a total neglect towards the subject matter to support the creative design process.

![Example comparative results using methodology](image)

Figure 4.20. Example significant quantitative results using the methodology
The qualitative results are highly context dependent. They range from observations to insights into behavioral patterns of the participants. Qualitative results are extremely important to illustrate and explain the quantitative results. Qualitative results tell a story and open a discussion for improvement of the educational tool.

Qualitative findings from a single participant might be as important to understand the educational process as the quantitative results of the entire pool of participants, especially when the pool is small. Participants might indicate the need for new functionality or indicate issues with the current available functionality. Qualitative results enable the evaluation of the importance of the introduction of a subject matter to designers and evaluates how the educational tool should be improved to enable this further.

<table>
<thead>
<tr>
<th>U</th>
<th>C</th>
<th>A</th>
<th>EXPLANATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>=/+</td>
<td>+</td>
<td>The best case scenario.</td>
</tr>
<tr>
<td>+</td>
<td>-</td>
<td>+</td>
<td>An increase in both the understanding and application with a decrease in creativity suggests a fixation on the tool, or a learning curve in the introduction of the subject matter, leaving no room for creativity. It can also indicate the incoherence of the tool within the design process. The distinction between these three should be evaluated in the concluding interview.</td>
</tr>
<tr>
<td>+</td>
<td>=/+ =/-</td>
<td>An increase in understanding that cannot be applied in the designs indicates a lack of experience from the participants view. More research is required on a longer term to evaluate the effect of the tool.</td>
<td></td>
</tr>
<tr>
<td>+</td>
<td>-</td>
<td>=/-</td>
<td>An increase in the understanding that both cannot be applied and restricts the creative process indicates a fixation. This fixation can either be due to the introduction of the subject matter, or the introduction of the specific tool in general. This is more likely when the educational tool is a software tool (Landay, 1996).</td>
</tr>
<tr>
<td>=/-</td>
<td>=/+</td>
<td>+</td>
<td>The lack of understanding, but increase in application suggests mimicry of the example behaviours. The increase in creativity however suggests an explorative and creative use of the tool, enabling to deduct interactive qualities opposed to merely mimicking them.</td>
</tr>
<tr>
<td>=/-</td>
<td>-</td>
<td>+</td>
<td>The lack of understanding suggests that participants resorted to mimicry of the available example behaviours that are appropriate, but lack authenticity. This too can be evaluated using the interview.</td>
</tr>
<tr>
<td>=/-</td>
<td>=/+ =/-</td>
<td>Solely an increase in creativity might be due to the exploration and inspiration induced by the tool, or by the purposeful neglecting of the troublesome subject matter. The distinction by this measure should be evaluated using the concluding interview.</td>
<td></td>
</tr>
<tr>
<td>=/-</td>
<td>-</td>
<td>=/-</td>
<td>The worst case scenario.</td>
</tr>
</tbody>
</table>

Figure 4.2 | The qualitative explanations for quantitative results
Results

Limitations

Quantitative
  Understanding
  Creativity
  Application

Qualitative
  User Interface

Summary
This chapter presents the limitations of the research, the resulting quantitative and qualitative results and summarizes the results for evaluation. These results are used for final evaluation in chapter 6. Evaluation and recommendations in chapter 7. Recommendations.

LIMITATIONS

Participants are limited to design students from the Industrial Design Engineering faculty of the Delft University of Technology. The design context is limited to a single design brief. A single design brief offers no guarantee for other contexts.

The current representation of educational tools for designs within the context of distributed networks is limited. The base line for comparison is a test case based on the bare minimum functionality present in Chopper. The variability in the introduction of new technology to designers with a new tool evaluated with a new methodology is very large. The outcome of the research is limited to making suggestions rather than conclusions. To increase the potential of the design and the research and limit the variability as much as possible, the following measures are taken:

• Testing of initial assumptions early on using self-evaluation, small user tests and pilot studies
• Minimization of inter-personal variability based on skills and experience using persona categorization
• Testing and optimization of the design brief ensure designers are inspired

A risk with using electronic tools is fixation on a single design rather than exploring several ideas at the start of a project (Landay, 1996). However, an electronic tool might as well provide valuable insights empowering the participant (Fischer, et al., 1994). This means that the use of an electronic tool might go two ways. Both of them need to be taken in the consideration.
5.1. Quantitative

This section presents the quantitative results according to the criteria of the methodology: the understanding, the application and the creativity in the base line test case and the test case that presents the design proposals. As mentioned in the chapter 4. Methodology, the quantitative evaluation of design proposal 2 is unreliable and not presented in the quantitative results. Solely design proposal 1, as presented in 3.2. Hypotheses | Design proposals is evaluated quantitatively.

5.1.1. UNDERSTANDING

The metric for the understanding of distributed networks is:

1. The grade from 0 - 10 from the subject matter related exam

Each participant completed the same written exam that evaluated the 6 facets of understanding in the context of distributed networks. The average grade of each test case is presented in Figure 5.2. The average grade of test case 2 is considerably higher than test case 1. Figure 5.3 presents the individual scores for each participant grouped with a participant on similar experiences and skills as presented in the methodology. The individual scores show an overall higher score for participants in test case 2: hypotheses. Participant 8 – in the hypotheses test group, participant group 3 – finished the exam in 10 minutes instead of the given 15 minutes. This may have caused the relative low grade within the participants in test case 2.

Following citations from the written exams illustrate a low scoring (first) and a high scoring (second) explanation of a distributed network:

“Components ‘interact’ with each other without interference of humans. Human = only input. DN communicates in a smart way, but what way I can not remember” - participant 6.

“A network of entities (nodes) in which all nodes are connected to each other, but not directly (in all cases). Each node is only connected to a certain amount of neighbours, which are in turn also connected to neighbours and so forth. This allows for the creation of systems that aren’t driven by a single entity, but has each node follow a certain set of rules to display a certain type of behaviour. This is easily scalable when the systems need to expand” - participant 7.
5.1.2. CREATIVITY

The criteria creativity is measured in the following metrics:

1. The increase in interaction designs
2. The increase in innovative interaction designs
3. The increase in most innovative scoring design
4. The increase in authentic designs
5. The increase in different designs

Figure 5.4 illustrates the average number of designs per participant for each test case. The minimal difference between the test cases indicate the need for minimizing inter-participant variability.

The average increase in Figure 5.5. Average increase in the number of designs per participant between test phases in the number of designs is expected to be negative or zero, inherent to a design process where creativity decreases with time. Participants in the hypotheses test case displayed a continuation of their creativity, whilst participants in the bare minimum test cases indeed show a decrease in the average number of designs produced.

Similar to the average number of designs per participant, the average innovativeness of each group of participants within each test case contains too much variability to compare. Figure 5.6 presents the average increase in innovativeness between phases for both test cases. The increase in innovativeness shows an added difference of one between both test cases on the Likert scale.

To illustrate the differentiation between an innovative design and a not innovative design, Figure 5.8 presents a design that was evaluated as not innovative (strongly disagree on the innovativeness Likert scale) and Figure 5.9 presents a design that was an innovative solution to the design brief (strongly agree on the innovativeness Likert scale). The non-innovative design is a pixel display that does not interact with users on the square. The innovative design however presents an interaction that responds differently to different users. The latter design invites users to explore the potential and connects users across the square. It answers the design brief and is a good example of an innovative design.
Figure 5.10 presents the average increase in the highest innovative scoring design per participant. The difference between the test cases is in the multitude of 0.7 on the Likert creativity scale. The highest scoring design is a more representative metric for creativity during the design process as many bad designs are necessary to converge to an innovative design. These designs influence the average score of innovativeness, but don’t influence the metric for the highest scoring design.

Figure 5.12 presents the increase in the number of authentic designs between phases per test case and Figure 5.11 the number of different designs produced by each participant. The authenticity of participants in both test cases is similarly increased, indicating that participants get more authentic with time independent from the use of Chopper. Participants in the bare minimum test case are able to diverge the produced designs, illustrated by the average increase in the different design per participants. Participants in the hypotheses test case however show no difference. This suggests a fixation on either the example behaviors or a fixation on properly applying the interactive qualities of a distributed network. The results in the application criteria should be able to differentiate between this.
The inter-rater variability is determined to evaluate the overall reliability of the ratings – if the ratings between the experts contain too much variance, the overall rating become less reliable. A first indication of the variability among the ratings is the overall bias of each expert on the Likert scale: the distribution of the ratings across the scale.

Figure 5.13 shows a bias of expert 1 towards overall agreement. Expert 2 however shows an overall bias to disagreeing on the innovativeness of most designs. Expert 3 shows a natural spread of ratings. Expert 1 is the coordinator of the course Interactive Technology Design at the Delft University of Technology. The role of coordinator might cause a bias to an overall compliance with the initial concepts due to expected improvement in future iterations. Expert 2 is least experienced in the evaluation of designs, which might cause the overall bias to an overall noncompliance.

The variance between ratings is shown in Figure 5.14. A variance higher than 1 means that experts disagreed on average more than a single multitude on the Likert scale. The variance in the ratings is very high, as was expected for a subjective measure such as innovativeness. The histogram displays the frequency of the variances in the multitudes of the Likert scale.

To illustrate the reasoning behind this variability, Figure 5.16 illustrates a design with low variance and Figure 5.17 a design with high variance. The left design illustrates a clear game that uses both the interactive tiles on the square as well as the building facade. The context is limited and has little room for interpretation. All experts rated this design as agree on the Likert scale. The design on the right however was ranged between strongly disagree, agree and strongly agree. While this design elaborates on the functional behaviour, it doesn’t explain the global behaviour of the system. It leaves much room for interpretation, causing the high variance between ratings.

Variability induced by leaving such wiggle room for interpretations may be reduced in future works by refining the design templates. A clear interactive behaviour description will reduce the variance in interpretation greatly.
5.1.3. APPLICATION

The application criteria is evaluated using the following metrics:

1. The increase in the number of applied designs,
2. The increase in the average application and
3. The increase in the highest scoring design.

Figure 5.18 shows the increase of the total number of designs per test case that score a 4 or 5 on the Likert application scale with a difference in a multitude of 2 designs. This suggests that while participants have been able to produce applied designs in phase 1, they are unable to produce new designs over a longer period of time without the added functionality in the tool. The initial ability to develop applied designs might be due to reproduction of the example behaviour sketches in phase 1.

Figure 5.19 presents the average increase in the average score on the Likert application scale. Figure 5.20 presents the increase in the average highest scoring design on the Likert application scale. These results suggest that indeed participants in test case 2 produce fewer designs that are authentic to focus on the proper application of the interactive qualities of a distributed network. The average score differs on a multitude of almost 2 on the Likert scale. This suggests that indeed, the participants with added functionality have shifted focus to applying the subject matter at hand. While participants using the bare minimum version of the tool show little difference in the highest scoring design on the application scale, participants in test case 2 show an average increase of almost a multitude of 1 on the Likert application scale.

Figure 5.21 is a design that did not present the interactive qualities of distributed networks (strong disagree on the application Likert scale). Figure 5.22 is a design that does (strongly agree on the Likert scale).

Figure 5.21 illustrates an interactive museum where individual elements of the square react to users by displaying information. The functional design of this network does not use any networking. Each individual element might as well be a standalone entity; hence it does not use any of the interactive qualities of a distributed network. The design on the right displays a ripple effect when users walk across the square. This ripple effect is inspired by emergence as seen in bodies of water in nature and is key to the emergent behaviour that distributed networks display. Each tile is interactive, but distributes the local interaction to generate an overall global ripple effect.
Opposed to innovativeness, the application of interactive qualities in a distributed network is an objective measure and should contain limited variability.

A first indication of the variability among the ratings is the overall bias of each expert on the Likert scale – the distribution of the ratings across the scale.

Figure 5.23 shows a strong bias for the distributed networks expert 1 for the agree category on the Likert scale. Compared to the other experts, expert 1 might have been too compliant in scoring the application. Expert 2 shows a higher bias against the neither agree nor disagree category. This suggests a difficulty in understanding the application of the designs. Solely expert 3 shows a distribution along the Likert scale. This indicates his experience in distributed networks and the need for more explicit evaluation guidelines for less experienced experts.

Figure 5.24 shows the overall variance in the ratings. To agree on the “official” rating of the application – so all agree or all disagree, a maximum variance of 1 is allowed. A larger variance suggests a disagreement on the “official” rating of the application. The overall variance in the ratings amongst all experts is too high to provide reliable measures. Inspired by the difference in bias of the raters, the variance amongst rater 2 and 3 is determined as well.

The variance between expert two and three is displayed in Figure 5.27. The most designs are rated with a variability less than one, indicating an overall agreement between these experts.

Figure 5.26, Figure 5.28 and Figure 5.29 present the quantitative results for the application metrics by all experts (1) and by solely expert two and three (2). Especially the average increase in the highest scoring design for interactive qualities shows a big change. It displays that participants using the full version of Chopper show a difference in the increase of application in a multitude of 1.5 on the Likert scale.
5.2. Qualitative

This section discusses the qualitative results that arose during the explorative comparative pilot study to evaluate Chopper. These results were deduced from interviews, observations and out-loud thinking.

The randomness in the interactive environment in example 1 was confusing to all participants. The liveliness of the system without direct user input was often seen as unwanted, unpredictable or entirely random. Most participants visually detected the gradient in the mowing of the grass, but no participant understood or detected the gradient spread in the growth of the grass. The gradient spread of the grass without user intervention in a randomized behaviour made the overall interaction with the system too disconnected from the users. Similarly, most participants displayed difficulties in understanding the syntax: distribute settings, toggle and global settings were often misunderstood in their functionality.

Participants requested the possibility to alter network configurations to change the overall behaviour of the system. This indirectly suggests a request for the possibility to alter distribution patterns. This suggests the misunderstanding of the distribution throughout the system. A distribution pattern can be predefined with local rules without altering the connections between neighboring tiles.

Programmers were more likely to ask functionally detailed questions. Participant 7: “Do the tiles know their position in the network?” These questions suggest a higher understanding of the interactive qualities of a distributed network. Subsequently, participant 7 was one of the only 2 participants that received a passing grade for the understanding exam. The other participant was participant 5, incidentally also a programmer. Participant 7 and 4, the most experienced programmers in the demographic explicitly asked for alteration of the behavioral code. Participant 7 even wanted to see the actual code opposed to the visually abstracted code presented in the local view.

Designers however, were more likely to add many users and change perspective views to get inspired. They requested the possibility of changing appearances. Such a functionality would be low effort, high reward functionality and might be interesting in future works. Participant 8 got “too” inspired by the behaviour in the examples and interpreted random behaviour as intentional. This blurred his view of the capabilities of a distributed network and explains the second to lowest grade in the pool of participants. Both participant 8 and 9 interpreted the notion of distributed networks as a conceptual model in the explanation facet related question in the written exam:

Participant 9: “A distributed network is a concept where individuals (people & objects) form connections between each other, with the use of sound, light or movement. This makes you more aware of the environment that you’re in. Which can be for more safety, fun or orientation.”

These explanations of a distributed network, taken from the understanding exam indicate the user-centered approach these participants take in understanding this technology. While they are not wrong, both the explanations miss the essential technological explanation of the functionality of a distributed network.

Participants in test case 1 using the bare minimum version of the tool were easily bored after phase 1. While this might have been expected, it might have also influenced the quantitative results. Participants mentioned that their boredom caused them to derive inspiration mostly from the design brief. As time passed until phase 3, inspiration from the brief was depleted. This caused the participants to be even more creative in using the tool. More research into using the tool over longer periods of time should indicate the effect of the design proposals.

Participants in test case 2 were however eager to explore realizations of concepts they had developed. This explains the increase in the application of the interactive qualities of a distributed network. This might also potentially cover the lack of difference in the number of designs between concepts generation phases. Designing a detailed design takes more time and more effort and limits the participants to generating a similar or smaller number of designs.

Some participants in test case 2 showed difficulty in selecting which local settings to change and distributed throughout the network. The initial experience of the tool was explorative, but this changed over time to restrictive as participants got bored or lost. The added control with the local settings interface enabled the participants to develop more innovative designs. These designs required more effort to produce, but had an added dimension to it due to the application of the interactive qualities.

Participant 8: “A distributed network is a set of laws, principles and behaviors that are given to an environment (tiles) to react to certain inputs. The context/users/... are crucial for this because they explore and shape their interactions, executed to the act of laws given.”

Participant 9: “The incremental addition of functionality in the hypotheses version of the tool matches the design process and forces you to understand the system one step at a time.”
5.2.1. OVERALL INTERFACE

This section presents some interface specific insights and issues that arose during the use of Chopper chronologically.

The example library should present a more thorough explanation than merely a name. Participants did not know what the examples’ purpose was and requested more information. The first phase of the procedure is almost spent entirely on exploring the navigational tools. Especially gamers, participants that compared the navigation to computer gaming navigation, had difficulty in accepting the basic navigation tools in Chopper.

The visualization of the on/off state of the tiles turning black was often misunderstood as being output color related. Participants either didn’t understand the meaning of the word toggle, leaving them to guess its functionality. Participants expected the toggling to impact the visible connections in the selected tile. As Chopper doesn’t visualize the active and passiveness of the connections, participants were left disappointed and confused.

The random percentage slider’s purpose wasn’t clear, as it showed no direct effect. The random button would need to be integrated in the slider’s functionality. The drawing panel’s potential of turning other variables on/off was only understood by participant 7 and 9. Both participants used this functionality in their designs.

The alteration of local settings did not show any direct effect in the system. The need for a distribution through the network was unclear as the distribution button was too physically and visually disconnected from the settings interface. A more realistic representation of the local settings is visualizing any changed in the settings solely in the selected node. This visualizes the current system status as is required in good user interfaces and suggest the need for further actions if the user wants the entire network to change accordingly.

Finally, participants indicated the need for assigning specific behaviour to players in the simulation to emulate specific use cases, e.g. all players walking to the center of the square at once, players chasing after each other, walking in a straight line etc. These use cases would suggest more exploration of the behaviour examples sketches.

LOCAL VIEW

The participants didn’t show any interest in the visually abstracted code. This was either because the participant was overwhelmed by the added functionality in the screen, or the abstraction from the actual code. Instead, participants were more interested in “discovering the unknown” - the distribution patterns that they were able to visually see and modify.

The visual presentation of the distribution pattern was however difficult to understand as it required stepping through the code. Participant 8 decided to skip the local view entirely to focus on understanding the global-to-local behaviour using the local settings interface. The other participants explored the local view a minimum amount of time before turning to the global view. This suggests that the current design of the local view does not match the designer’s skills and experience and requires another design iteration.

The methodology provides quantitative metrics to enable comparison of the results. To summarize the effect of the first design proposal, each criterion is represented by an average of all its metrics, presented in Figure 5.30. This representation is a false representation, as the metrics require weighing, but provides a quick overview.

The understanding rating represents an overall concluding evaluation for both groups of participants. Both test cases show failing grades on average. However, the only 2 participants that received a passing grade were in the hypotheses test group.

The creativity and the application criteria are evaluated using the increase in between test phases and use a different scale than the grade scale of the understanding. The difference between the creativity metrics is minimal and requires a detailed evaluation as is presented in the next chapter. Qualitative results and the combination of the creativity and application metrics however suggest that participants in test case 2 showed a fixation on the tool limiting their creativity.

Subsequently, the application score of the designs is higher than the participants using the bare minimum version of the tool. This suggests that a limited increase in understanding. This increase also covers the lack of significant increase within creativity: participants were focused on applying the interactive qualities opposed to being creative.

![Average criteria results](image)

Figure 5.30 Summarizing the quantitative metrics of the evaluation without weighing factors
6 Evaluation

Methodology
- Research setup
- Criteria

Chopper
- Quantitative
- Qualitative
- List of requirements
- Distributed networks for designers
6. Evaluation

This chapter evaluates the thesis work. It evaluates the educational effect of Chopper using the results derived from the explorative comparative pilot study. It evaluates the overall success of the tool with the prior defined list of requirements. Subsequently, the introduction of distributed networks to designers is evaluated through the success of chopper.

The chapter also evaluates the evaluation methodology structure illustrated with the explorative comparative study used to evaluate Chopper. Finally, it evaluates the criteria used to quantify the results in the methodology.

Chapter 7 presents the recommendations derived from these evaluations.
6.1. Methodology

This section evaluates the methodology using the pilot study evaluation of Chopper. It evaluates the setup of the study and the evaluation methods for the methodology specific criteria: understanding, application and creativity.

6.1.1. Research Setup

Participants

8 participants are categorized prior to the research. Each participant was categorized in a group with another participant based on the skills and experience. These participants were then spread over the 2 test cases to have a representative demographic of the total participant pool for each test case.

The participant sample size will need to be a lot larger to limit variability enough to provide significant results from the quantitative research. The sample size formula presented in Figure 6.2 uses the following properties to determine the required sample size:

1. The power: the certainty of detecting the difference. Generally this is 90%, but because the evaluation is also explorative, the minimum quantity of 80% is recommended.
2. The significance level of the difference criterion – the cutoff below which the null hypothesis is rejected is generally p = 0.05, or 5%.
3. The effect size (d) – the smallest difference in means that would indicate a meaningful difference.
4. The standard deviation (SD) of the outcome.

To calculate the recommended sample size: participants per test case, the standard deviation is taken from a similar research. The standard deviation for the creativity and application measures is difficult to find in a similar research, however, the understanding deviation can be taken from an exam. Instead, the results of the final Programming in C exam are used – an introductory programming course for first year computer science students from the Delft University of Technology are taken. A histogram of the exam results is provided in Figure 6.3. This course introduces the concept of programming, however participants have been able to study intensively opposed to the context of using Chopper.

The meaningful difference that should be detected is an effect size of one between the average grades of the group. This means an average difference of at least one point in the total grade. The standard deviation of the grades in the programming in C course is 1.88. To measure the effect size for a significant difference of 1/10th of the participants' scores, a sample size of 55 participants is required for each test case. If the difference appears to be less than 1, the null hypothesis is confirmed. If an effect size of 2 is calculated, the sample size for each test case is merely 14 participants.

Inter-persont results of the designers, makers and programmers can be compared when each test case contains a sample size of 14 participants for each persona, a total of 84 participants. Additionally, categorizations of persona should be based on explicit measures, by means of a questionnaire and an explicit test to evaluate the skills and experience. If the participants are categorized correctly, the metrics of the methodology can be measured as a total instead of the increase in between test phases.

The participant pool in the pilot study is too small to draw real conclusions. The results of some participants suggested previous experience in using distributed networks, while this wasn’t shown in the previous questionnaire. This indicates the need for better formulation of the questions in the questionnaire as well as the addition of an explicit test.

However, the pool of participants proved to be adequate for a pilot study and didn’t show any outliers in the quantitative results.

Figure 6.3. Histogram of the Programming in C grades

Figure 6.4. Mean difference in the understanding exam

---

1 EE1400 - Programming in C // Computer Science Bachelor course exam. - Delft University of Technology, 2012.
DESIGN TEMPLATES

The templates provide a framework to limit any variability that is induced in the representation of each design for different participants. Each A4 template presents a designated space for the global behaviour within the context of the design brief, network communication and a description of the design brief. Persona stickers and colored pens are provided to fill the designated spaces.

A design template limits the variability induced by drawing skills, representation of the context and the level of detail in the application of the subject matter as well as answering the design brief.

Resulting designs illustrated much variability in the explanation of the application of the distributed networks. All participants that used the bare minimum version of the tool had little idea in how to represent each autonomous tile in the designated network area. The provided template presenting the 9 tiles subsequently primed the participants in how to explain the network communication. Priming should be limited to the use of the tool. Providing a generic grid will perhaps lower the correct application of the distributed networks, because of this the different interpretations of the application of distributed networks induced during the use of the tool will be better visible in the produced designs.

The design templates limited the variability induced by the drawing skills of each participants and the representation of the context. The setup presented a nice overview and allowed for a fast comparison using both the visual representations of the global and the local interactions.

TEST CASES

The test cases in the research compared the design proposals with a bare minimum version of Chopper. The bare minimum version of Chopper emulated the current context of available educational tools in the context of distributed networks in design. The second test case added functionality inherent to the design proposals incrementally in each phase. Each incremental addition is evaluated against the bare minimum version of the tool.

Test cases represent the individual methods deployed within the educational tool to evaluate the effect of each individual method or technique independent from others.

The bare minimum version of the tool is representative for the current context of educational tools as this context is virtually inexistent. As mentioned before, the incremental addition of design proposal 2 to design proposal 1 in test case 2 does not provide reliable quantitative results. The caused effect may be due to a longer period of time experimenting with the first proposal or the addition of the second.

The test cases enabled the quantitative evaluation of design proposal 1 versus the bare minimum functionality of the tool. Inherent to the size of the pilot study, the test case presented enough results to draw suggestions and make recommendations for future work. The second design proposal is evaluated using the qualitative measures.
6.1.2. CRITERIA

UNDERSTANDING

The understanding criteria evaluates the 6 facets of understanding using a written exam format. This exam is presented after the entire research to evaluate the understanding of each participant. The exam scores are evaluated based on a score sheet developed prior to the research. The overall grade of each participant is used as the indicating quantification of the understanding criteria.

As mentioned before, when participants are categorized adequately in large sizes, the overall measure of quantification can be used to indicate the criteria. As the participant pool was fairly small during the pilot study, inter-participant differences may induce variability. The initial assumption that participants did not have any prior knowledge, experience or understanding in the subject of distributed networks may have been incorrect. Any variability amongst this criteria may give participants an unfair advantage and present itself as a result from using the tool. The pilot study should have instead evaluated the exam beforehand and the exam afterwards. However, as participant 3 explained, the exam enabled the participants to evaluate the understanding that is required to design for these networks. Instead, the pool of participants should be increased and categorized to minimize the inter-participant variability.

Take notice that other subject matters that are already familiar to designers may need a different approach. In these subject matters, no assumption can be made on the initial skills and experience of the participants. Instead, participants should be selected and categorized with care based on these properties.

The questions in the exam were essential questions, chosen with care. A large number of questions will relatively reduce the variability. An evaluation of the understanding questions is made. The average score for each question is illustrated in Figure 6.7.

Question one, two and six show fairly straightforward results: participants using the tool with hypotheses are better able to explain, interpret and evaluate their self-knowledge in the context of distributed networks. Question 6, related to self-knowledge asks for a self-evaluation of the produced designs. It can be argued that this question is not related to self-knowledge, but more to the application facet of understanding. If so, the participants show a relative increase in the application, opposed to the other application facet question 3. To evaluate participants’ self-knowledge, it may be more appropriate to ask for an evaluation of the other 5 facets of understanding:

You’ve answered 5 questions related to explanation, interpretation, application, perspective and empathy within distributed networks. Please rate your own answers and explain why.

Participants in the bare minimum test case illustrate better scores in question 3: application. This suggests that the bare minimum version of the tool allows the participants to look beyond the example behaviors where as participants using the full version might have been fixated on the example behaviors. However, more research in interviews should actually illustrate whether this is true.

All participants in the bare minimum test case were unable to answer question 4, related to the perspective facet. The perspective facet of understanding is related to the motivation to design for a distributed network. This suggests that the limited functionality of the tool does not motivate designers to design for a distributed network. This may due to [the feeling of] limited control in a distributed network.

Almost no participant has question 5 regarding empathy correct, suggesting it was too difficult. This may have been to the lack of knowledge in the subject matter in order to come up with properties than can be evaluated as strengths or weaknesses. Since this question should evaluate the understanding instead of the knowledge, it may be more appropriate to ask for an evaluation of the knowledge.

Which of the following properties are strengths and which are weaknesses of designing for a distributed network? Choose at least 2 for each: scalability, modularity, safety, reliability, performance, costs and control.

Further research would need to indicate the right syntax to use within this question.
CREATIVITY

The creativity is measured through expert evaluations on a Likert scale to evaluate the innovativeness of the design and similar sorted designs.

Creativity and innovativeness are abstract notions with many definitions throughout literature. Similar to the understanding, many and many different assessments are required to assess such notions.

The experts showed high variability in evaluating the creativity. This indicated the many different views an expert may have regarding the innovativeness of a solution. When many experts are used with high variability in ratings, every design will average around the middle of the Likert innovativeness scale. It can be argued that the evaluation guidelines should be more focused in order to limit this variability. A focus group study with all design experts is required to evaluate whether this would make sense. The same focus group could be used in order to define a new non-ordinal scale for evaluation.

APPLICATION

The application is measured through expert evaluations on a Likert application scale to evaluate the application of the interactive qualities of a distributed network. The criteria is characterized by the number of applied designs, the average score of application and the highest scoring design in application. These ratings are evaluated by experts within computer science fields, primed with the redefinition of distributed networks.

The application criterion differentiates the designs that flourish when developed with a distributed network and the designs that would suffer from this topology – in reliability, responsiveness or overall interactive qualities. This distinction between suffering and flourishing is in some sense the innovativeness of applying a distributed network. Combined with an overall innovative interaction designs, these designs have the potential to become technology epiphanies.

Many designs presented limited details in the functionality of the system. This caused variability in interpretation by the experts. Experts did not agree on whether the design could apply the interactive qualities or whether it actually shows that it applied the interactive qualities of a distributed network. This variability should be limited after redesigning the design template.

Experts that are experienced in the development of interactive distributed networks show a better distribution in the evaluation the application of the designs. Other experts that are not as experienced show more interpretative errors. Additionally, focused evaluation guidelines should be provided to explicitly explain the interactive qualities of a distributed nature, focusing on the autonomy, communication and collaboration. More priming examples can be used to prime the experts on the entire scale.

This interaction design applies interactive qualities of a distributed network:

<table>
<thead>
<tr>
<th>Strongly disagree</th>
<th>Disagree</th>
<th>Neither agree nor disagree</th>
<th>Agree</th>
<th>Strongly agree</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
<td><img src="image3.png" alt="Image" /></td>
<td><img src="image4.png" alt="Image" /></td>
<td><img src="image5.png" alt="Image" /></td>
</tr>
</tbody>
</table>

Figure 6.8. Priming over the entire Likert application scale
6.2. Chopper

6.2.1. Quantitative

Quantitative results allow for an evaluation, but are not reliable enough to draw conclusions due to the size of the performed research. Additionally, the research is an explorative comparative study that compares 2 test cases—one with a bare minimum functionality of Chopper and one with the added functionality inherent to the hypotheses. This setup evaluates the difference between different versions of the tool, but doesn’t allow for a comparison of the current educational context of distributed networks in designs. This context is inherently inexistent as there are no other tools or methods yet available that introduce distributed networks to designers. To compare the tool against such a context, a baseline representation will have to be set up. The tool could be evaluated against the already available general and central design tools such as sketching.

The added functionality in the second test case shows a relative increase in both the score for understanding and application of distributed networks. Participants in test case two are better able to apply the interactive qualities of a distributed network, but score similar results in the creativity metrics. These participants are unable to answer the application facet of the understanding exam. This indicates either the success of the explorative experiential nature of the tool that allows participants to apply what they see and iterate on this or perhaps a mis-formulation of the question.

Inherently to the design context, it would have been optimal if the tool supported an increase in creativity. Participants that used the version of the tool with added functionality indicated that the designs costs more effort to produce over time. More research will need to indicate whether more experience will require less effort to produce designs, enabling designers to diverge and be more creative.

6.2.2. Qualitative

Initial use of the tool was experienced as explorative, but became restrictive over a longer term. This was due to the finite amount of examples and the little explanation provided within the tool. More research will need to indicate how much explanation is appropriate in such an educational tool. More research in the interface and textual design of the educational tool is required in order to further tailor the tool for designers.

The research suggests that participants were fixated on the tool and its example behaviours initially, which reduced over time. Further research will need to indicate whether this fixation over time decreases enough to support the creative nature of the design process. This is expected to occur due to the nature of threshold concepts. Once a threshold concept is understood through a conceptual gateway, Chopper in this case, participants will be able to use explain, interpret and apply these concepts.

The incrementally added functionality as presented in test case 2 was positively received. It allows the participants to educate themselves step-wise and prevented them from feeling overwhelmed.
6.2.3. LIST OF REQUIREMENTS

RECOGNITION RATHER THAN RECALL
The simulations of the example behaviours in Chopper represent a deployment of a distributed network in the real world. Users can interact with the system and navigate through the environment. Each tile is autonomous and connected to neighboring tiles. Each individual tile can be individually accessed to alter settings as well as initiate a distribution of new settings throughout the network.

Not all syntax was recognized, syntax needs another design iteration to ensure the designers recognize these interface elements.

VISIBILITY
The status of each individual tile is always visible, except when local settings are altered. These alterations should be directly visible to present any confusion. The overall visible functionality in the interface is kept to a minimum and is less than 20 functions. The incremental increase of functionality through traversing between alternative views in the system is positively received and supports the nature of the design process.

REFERENCING
Reference sheets were provided during the research to indicate the available functionality within each view. Participants did not understand the initial behaviour of the examples and requested more information. Inspiration can be taken from the info interface by NetLogo. Each example behaviour should contain information about its purpose, its functionality and means for expanding the functionality.

EXPERIENCE
Participants could not explore use cases that needed control over several players in the simulation. The inability to steer players increased their interpretation of randomness in the interactions.

CHOPPER
Existing example behaviors allow users to hack settings to incorporate new functionality within 5 minutes. The settings are pre-selected and require minimum effort to change. Two behaviour examples are available. The development of more behaviour examples is highly encouraged, but not executed due the time limit.

Examples presented a minimum of 5 settings to change in order to explore the different settings. A profound understanding was necessary to choose an interesting setting for behaviour. It would be easier if combinations of interesting settings were presented in presets. This functionality would allow the user to compare alternative settings, which wasn't deployed in the preliminary version of the tool given the amount of time.

TRANSPARENCY
The transparency of the global-to-local programming is visible through the impact of individual states of the nodes on the global behaviour. The visualization of the distribution of new settings enables the participants to visually interpret the distribution.

The preliminary version of Chopper doesn’t support the modification of the configuration of the network, for example inputs and outputs.

AUTONOMOUS PHASE
Autonomous functionality will be deployed in future versions of Chopper. Currently, the topology of the network and the data distribution throughout the network is visualized. Each node illustrates its individual progress through the available outputs in the global view and debug mode in the local view.

Currently, no programming behaviour is functional. If Chopper expands, visual programming blocks should be provided to accommodate the additional programming of logic functionality, configuration of inputs and outputs and distribution patterns. Opposed to initial ideas of visual programming, the code should be presented in a similar fashion as the textual code in NetLogo. The designers that are able and interested want to have full access to the code. Those who are not, will be satisfied using preset alteration within logic functionality, configuration of inputs and outputs and the distribution patterns as seen in the local view.

LEARNING
Chopper presents recognizable modification tools that are fault-avoidant. They however do not support learning from mistakes. It does not allow for explicit verification of the behaviour, but presents implicit verification visible in the simulations. It does not allow for comparing different behaviors. This should be solved with the introduction of preset combinations of settings.

DISTRIBUTION
In the nut-cracking phase, distribution patterns should be made available. The local view in the tool visualizes the distribution of data in current behaviour examples. The local-to-local view presents the tools to alter and reuse functionality to develop new distribution patterns.
Chopper introduced a small batch of designers to distributed networks in the hacking phase of a design process. It has both inspired and motivated designers by enabling the exploration of the interactive qualities a distributed network has to offer. It allows for hacking of existing example behaviors by presenting settings that can be changed. This fault-avoidant exploration of the behaviors enables designers to understand and apply the interactive qualities in turn while remaining creative.

Ultimately, an educational tool for designers in the context of distributed networks should support designers throughout the entire design process. A final version of Chopper should support the development of autonomous solutions, nut-cracking of the software as well as actual deployment on autonomous nodes as a prototype. It should bridge the gap between distributed network experts and designers and open the floor for communication and discussion and isn’t meant to eliminate the need for experts entirely.

The preliminary version of Chopper solely supports hacking of a limited number of example designs. Designers should instead be inspired by a large number of example behaviors and favorably a large support community as seen in Arduino. A final version of Chopper or other tools supports designers throughout the entire design process – hacking, autonomy, nut-cracking, users and integration.

Chopper does enable designers in the applying of interactive qualities of distributed networks. Many designs depicted multiple individual nodes working together to create an overall global behaviour: global-to-local transparency. It enabled a fault-avoidant exploration of different behaviors within an example behaviors. By visualizing local rules, users are able to visually understand their effect. The autonomy of each node is displayed through the on/off functionality and the local settings alteration. It summarizes the autonomy and collaboration in the simulation.
7

Recommendations

Chopper

Introducing distributed networks to designers
Educational tools and methods in design

Methodology

Participants
Procedure
Test cases
Design template
Criteria
This chapter presents an overview of the recommendations made for improving the educational tool Chopper, further introducing distributed networks to designers and the development of educational tools and methods in the context of design.
7.1. Chopper

This section presents recommendations for improving Chopper and global methods and techniques that can be deployed within other educational tools for distributed networks in the context of design. The recommendations are based on the results of the explorative comparative study and the evaluation presented in Chapter 5 and 6 respectively.

1. Present more example behaviors in Chopper that are inherently distributed
   a. Token vs. gradient vs. spatio-temporal distribution algorithm
   b. Interactive environment vs. environmental interaction
   c. Random vs. systematic behaviour
   d. Research the adequate amount of explanation provided
2. Provide preset appearances for the nodes in the network
   a. Beam, tiles and balls with dynamic sizes
3. Present preset combinations of local settings to enable exploration of meaningful settings
   a. Add explanations for each preset
   b. Add direct visualizations of all settings that can be changed and distributed
   c. Enable visualized comparison of different presets
4. Enable dynamic network configuration by adding and removing nodes and connections
   a. Visualize the altered network connections
5. Support building behaviors from scratch
   a. Incrementally add functionality
   b. Visualize the functionality
6. Re-design and simplify the local view
7. Design the local-to-local view
8. Provide detailed behavioral rules for the players in the simulation to develop use cases
9. Provide much more information
   a. For examples
   b. For functionality within the tool
   c. Similar to the into-tab as presented in NetLogo
10. Research and refine used syntax specifically for designers
11. Research all design proposals using the methodology

7.1.1. LOCAL VIEW

The local view presents the “unknown”, the functionality of each autonomous tile that collaborates to present a global interaction and the distribution of data. The local view should function as a debug view that visualizes both the collaboration and the communication of the nodes in the system as presented in the previous design of the local view in Figure 7.2. The code is presented in the local-to-local view where further autonomous behaviour of the nodes can be defined.
7.1.2. INTRODUCING DISTRIBUTED NETWORKS TO DESIGNERS

This section presents recommendations in further introducing distributed networks to designers by any means of educational tool:

1. Support the incremental nature of the design process
   a. Allow designers to choose a level of detail to operate on, perhaps by enabling traversing
2. Emphasize the global-to-local opaqueness – visually connect the global behaviour to individual local behaviour
3. Present modifiable preset functionality through recognizable tools
   a. Present a wide variety of alterations that can be made in threshold constants, input and output mapping, but most importantly in the distributed algorithms provided
   b. Visualize present the settings
   c. Represent the need for global distribution of altered settings
   d. Visualize the distribution of these settings
   e. Present information about the settings
   f. Present preset combinations
4. Emphasize the distribution of data through the network
   a. Emphasize on collaboration and cooperation of the nodes in the network
   b. Enable individual state changes such as on/off functionality to experience the effect in the global behaviour
   c. Visualize messages send through the network
5. Motivate the use of distributed networks
   a. Emphasize scalability and modularity
   b. Present inherently distributed examples in a valid context

7.1.3. EDUCATIONAL TOOLS AND METHODS

This section presents some overall recommendations that can be derived from the process of designing an educational tool, or method.

1. Define the educational target audience
   a. Current knowledge and skills in the subject matter
2. Define the educational goals prior to the development of the tool
   a. Define an indication of when these are achieved
3. Choose and argument an appropriate form for the educational tool or method
   a. Software tools might induce a fixation (Landay, 1996)
4. Evaluate your tool during the entire design process with the audience
   a. Start with self-tests
   b. Use paper prototypes
   c. Use the methodology when a final or preliminary version is ready for evaluation
   d. Interpret the results and improve
7.2. Methodology

This section makes recommendations for improving the evaluation methodology for educational tools and methods in the context of design. It discusses each individual section of the methodology.

7.2.1. PARTICIPANTS

Collect enough participants for the goal of the research: pilot or full scale.

Determine the demographic of the participants prior to the research based on:

1. Explicit experience in subject matter related courses – grades and number of courses
2. Explicit experience in subject related tools
3. Explicit experience in the subject matter through describing a previous project that is most related to the subject matter and a small test in the subject.
4. Their primary and secondary design roles
5. Explicit degree information – bachelor and master

7.2.2. PROCEDURE

If the categorization of the participants is done in such a way to limit any inter-participant variability, measuring the increase in between test phases is no longer necessary. Instead, the overall measure for each (persona within the) test case will be a reliable measure. Tests merely require the same amount of time for each test case in order to compare the results. Still, it is recommended to divide the procedure into several concept generation phases to track the educational progress of the participants. This progress might be a proper indication for the learning curve and might unveil when the threshold concepts are overcome.

Figure 7.3. Recommended research procedure
7.2.3. TEST CASES

1. Determine the goals of the research:
   a. Prove that the educational method is better than current methods
   b. Evaluate the success of the methods and techniques in the tool
   c. Evaluate multiple methods to choose an educational method for a curriculum
2. Determine the base line comparison inherent to the goal of the research.
   a. Current educational methods and tools
   b. A bare minimum representation of your method or tool
3. Represent each independent method in an individual test case.

During the pilot evaluation, Chopper was evaluated with a bare minimum representation of the tool is used to evaluate the success of the design proposals deployed in the Chopper. To evaluate the overall success of Chopper by comparing it to the current available methods in the context of distributed networks, a base line should be set up. As this current educational context is virtually inexistent, go-to design tools are representative of the current available tools within the context of distributed networks (or any unfamiliar subject) within design. A central design tool is the process of sketching. The effect of the process of sketching can be used as a base line context to compare the effect of the use of Chopper against.

The variability between these completely different tools is extremely high. Normalizing the skills and experience of both tools in the initial phase of the research can reduce this variability. Examples for normalization are tutorials that guide the user through the process of iterating a distributed network using the tool at hand. To reliably compare this base line representation with the bare minimum representation of the tool, an equal amount of educational material should be communicated. Both tools should present the same examples with the same educational and communicational values.

Normalization through a tutorial should ensure that participants in both test cases continue the second phase of the research with the same level of skills and experience in both tools. Therefore, the normalization phase of the research should not be time-limited, but instead accomplishment-limited.
A design template is provided to structure the information about the produced designs to allow for an evaluation by experts. The design templates should however not prime the participants in any way. Therefore, a minimized layout should be provided that is independent from the actual subject matter. A realistic context illustration should be provided in a sketchy format to enable the exploration of the context through sketching. Stickers should be provided to limit the variability induced due to drawing skills.

Following recommendations are made to further limit any variability in the designs unrelated to the evaluation criteria in the methodology.

1. Explicitly designate space for each section to explain the section:
   a. What is it? (How does it answer the design brief?)
   b. What does it do? (What is the global interaction?)
   c. How does it work? (How does the network communication work?)

2. Limit priming through the design template by generalizing the subject specific designated answer space.
   a. Provide an abstract grid that can be used for displaying neighboring tiles as well as representing a notebook grid.

This section presents the recommendations for improvement of reliable results in each criterion in the methodology.

**UNDERSTANDING**
1. Add more exam evaluators that are not biased as the author is
2. Add more questions for less variability in the measurements
3. Choose questions carefully
   a. Are they essential?
   b. Multiple choice vs. open questions
4. Be careful with syntax and knowledge related aspects

**CREATIVITY**
1. Support the creative nature of the designers in the research
   a. Supply enough templates and pens
   b. Provide inspirational material
2. Use as many design experts as possible
   a. From different fields within design
3. Focus on evaluating the qualitative measures to explain quantitative results.

**APPLICATION**
1. Define and present the prove of application prior to the research
2. Prime the experts on all levels of the Likert evaluation scale
Conclusions

Research questions

Hypotheses

Research objectives

Concluding remarks

Note of the author
8. Conclusions

This final chapter concludes the thesis report by re-iterating key findings and conclusions during the thesis project.

It reflects on the initial research questions and evaluates the posed design proposals. It evaluates on the initial research objectives and finally evaluates the overall thesis with some concluding remark.
8.1. Research questions

WHAT PROCESS SHOULD BE SUPPORTED BY EDUCATIONAL SOFTWARE FOR INTRODUCING THE INTERACTIVE QUALITIES OF A DISTRIBUTED NETWORK TO DESIGNERS?

- An educational software tool should support the divergent and convergent iterative nature of the design process.
- The process of designing interactive qualities follows the hacking, autonomy, nut-cracking, user and integration phase.
- An educational software tool that introduces a subject matter to designers without prior knowledge or experience should support especially the hacking and autonomy phase of the process.
- The different design roles influence the goals of the designer in the design process.
  - A designer focuses on the development of concept designs and requires an explorative experiential process that inspires.
  - A maker focuses on the structural development of the designs and requires an incrementally explorative process that supports traversing through the global and local structure of the network.
  - A programmer focuses on the software development and requires full access to the distributed behaviour.

WHAT INTERACTIVE QUALITIES OF A DISTRIBUTED NETWORK NEED INTRODUCING TO DESIGNERS?

- The global-to-local opaqueness, the distribution patterns and their occurrence in nature are the qualities that are important to get designers inspired in the design of distributed networks.
- The scalability, the modularity and the robustness of the network are the qualities that are important to get designers motivated in the design of distributed networks.

HOW CAN AN EDUCATIONAL SOFTWARE TOOL FACILITATE THE INTRODUCTION OF DISTRIBUTED NETWORKS TO DESIGNERS?

- First and foremost, an educational tool can present many examples that are inherently distributed. The scope of examples should include interactive environments and environmental interactions, direct and indirect user interaction and gradient, token and custom spatio-temporal distribution patterns.
- By visually presenting all example behaviors with an introductory text, the overall scope of distributed network is apparent in a first glance.
- The educational software tool should allow traversing through the different views of a distributed network: global, global-to-local, local and local-to-local.
  - Supporting the user in freely experimenting with example behaviors through modification of local behavioral rules enables the user in understanding and applying the interactive qualities of a distributed network.

HOW CAN AN EDUCATIONAL SOFTWARE TOOL FOR INTRODUCING DISTRIBUTED NETWORKS TO DESIGNERS BE EVALUATED?

- A comparative study can be performed to compare to the overall success of the tool to the current available context of current educational methods. The same study can be used to compare the effect of independent methods and techniques deployed within the tool.
- The distributed paradigm has proven to be a threshold concept — a concept in need for a different way of understanding to progress in the subject matter. A first criterion to evaluate the tool is the change in understanding through means of a written exam.
- Inherent to design-driven innovation, a second measure to evaluate the effect of a tool specifically on the ability to design is the creativity of the designer.
- Finally, a third measure evaluates the effect of the tool specifically on the ability of the designer to apply the subject matter in his/her designs.
- Qualitative measures are however equally important to explain and evaluate the quantitative results and should be measured through interviews, observations and out-loud thinking.
8.2. Evaluation of hypotheses

1. Providing a settings interface to change the behavioural rules of demo projects similar to that of NetLogo will allow designers to explore alternative interaction designs. This quantitative exploration will enable the designers to be more creative in their design process.

The addition of the settings interface is the sole design proposal evaluated using the comparative study. The quantitative results suggest that the local settings interface increased both the understanding and the application of the interactive qualities of a distributed network. The creativity of the design however showed no significant results.

Participants indicated that the addition of the local settings interface provided them with the control to evaluate the application of the ideas they had in mind. This allowed them to focus on the application of the interactive qualities, allowing them to increase their understanding of the network and apply this in their designs.

2. By creating opaqueness in the topology, designers will be able to understand and apply the interactive qualities of a distributed network.

The transparency in the tool is visualized through the settings interface, but is emphasized through the visual representation of the distribution of settings. The individual changing of each node’s state also enabled the designers in how to apply this transparency as the opaqueness. Turning the individual tiles on and off helped the designers to understand how the data propagates if a single connection/tile is not working. The visualization of the transparency – or the opaqueness of the system is applied throughout the entire tool. The effect is appreciated by the designers and encouraged in future work. Note that the transparency should be clear – disabling the specific tiles should also visualize the disabling of their connections in the simulation.

3. Emphasizing the effect of communicating local autonomy enables the designer to understand and apply distribution patterns.

Visualization of the communicating local tiles is presented in the local debug view of the network. The overwhelming amount of detail in the local view buried any independent method and techniques, causing participants to skip this view entirely. Subsequently, no conclusions can be made on the effect of the emphasis on communication local autonomous node. Qualitative results indicated that participants were in fact able to make sense of the distribution through visualizations of the hardware connections. They were however not aware of the communication between these local tiles that are responsible for the overall distribution patterns. Continuation with the proposed redesign in section 7.1 of Chopper is encouraged.

4. By traversing between a global view of the system and an alternative local view of the autonomous entities, designers will be able to understand and apply the interactive qualities of a distributed network.

Similar to the third design proposal, the traversing between the global and local view does not allow for any concluding remarks due to the participants that choose to ignore the local view. The author still thinks that the deployment of both design proposals will help to transform Chopper into an actual conceptual gateway for distributed networks.

The incremental addition of details in the alternative views was explicitly appreciated by the participants. Similarly, incrementally traversing between these views will be very relevant.
8.3. Research objectives

Develop an educational tool that enables design students to explore, understand and apply the interactive qualities of distributed networks.

Chopper is an educational software that introduces designers to distributed networks through a multidisciplinary approach. The distributed paradigm is known to be a threshold concept and troublesome knowledge to acquire and apply within designs. The user-centered and technology-driven approach took the best of both worlds and as a first developed an educational software tool in the context of distributed networks specifically for designers. Quantitative research indicated a relative increase in the understanding and application of distributed networks in design when using the full deployment of Chopper. Further research will need to indicate whether Chopper is an improvement compared to the current available educational material context for distributed networks in design. While the quantitative research shows a relative increase, qualitative research has indicated that Chopper is not yet a conceptual gateway for the threshold concept of distributed networks. Users do not yet fully understand the distributed network concepts, merely 2 out of 8 participants scored a passing grade of 6.25 out of 10. Additionally, the designers are not yet experiences enough to innovate in the context.

The technology push of connected embedded devices has inspired and motivated designers to start designing for large scale interactions. Enabling designers in the design driven innovation of distributed networks has the potential to result in technology epiphanies. Participating designers in the research have shown great interest in distributed networks during and after the research.

Design molds and shapes the world and these technology pushes should not be limited to ad hoc applications in subject specific communities. However, designers merely need to understand and creatively apply these subject matters to develop design. Further deployment of the designs can be left in the very capable hands of these subject specific communities. In this case, experts in distributed networks.

Develop a methodology that evaluates the effect of educational tools and methods in the context of design.

The evaluation methodology allows educators and tool designers around the world to evaluate their educational methods and tool in the context of design. The methodology can be used to evaluate the overall performance of a tool versus a base line context, evaluate the individual methods and techniques within a tool or evaluate several tools and methods to select educational materials for a curriculum.

The methodology provides both quantitative and qualitative results that allows comparison. The methodology is merely tested and evaluated in a small explorative version of the setup. The quantitative results offer no absolute measures, but instead should be used as a framework for evaluation. Qualitative results are at least equally important as they provide explanation of the quantitative results and indicate suggestions for improvements.

The methodology is recommended for use during the entire design process of a tool to evaluate the effect of all methods and techniques. The research size can differ according to the phase of the design process to both a small sized pilot study and full-scale research.
8.4. Concluding remarks

Chopper introduces distributed networks to designers in the hacking phase of the design process. It enables designers to bridge the gap between design and distributed networks. Inherent to the initial explorative vision of Chopper, it provides designers with the tools to explore the interactive qualities of a distributed network. Chopper presents a limited global representation of the distributed network by abstracting local details. The global view represents a hot air balloon, enabling a global perspective, opposed to the detailed exploration initially envisioned with a helicopter, or Chopper.

After designers have been able to understand and creatively apply the interactive qualities of distributed networks in their designs, domain experts are still required for the final deployment of the systems. Designers will never replace the subject specific experts, but are able to expand the horizon by diverging the applications of the technology through design driven innovation.

If designers were to continue in the use of distributed networks, more tools are required to enable the prototyping of these networks. The prototyping of these interactions will allow designers to further test and iterate the designs.

8.4.1. NOTE OF THE AUTHOR

Writing this thesis has been an exploratory experience for me too. It has been with great pleasure that I was able to combine both my love for design as my love for embedded systems in my final master thesis. I initially thought I already knew the best of both world through education, but actually applying both fields into a single project has been an eye-opening experience.

I have learned so much about the understanding and researching during the development of the evaluation methodology. Never would I have imagined that my final thesis would have such a strong research orientation, but I believe it has been vital in the evaluation of Chopper and the continuation of my work: introducing technology to designs. I strongly believe that designers shape and mold the world and I think it is vital to share expertise across fields to innovate. Innovation is apparent in every subject field and sometimes all it needs is a different perspective.
Bibliography & Appendix
Bibliography


Dulay N., Kramer J., Magee J., Sloman M. and Twindle K., "Distributed System Construction: Experience with the Conic Toolkit".


EE1400 - Programming in C // Exam. - Delft : [s.n.], 2012.


Pauling Linus A generally quoted comment.


Referenced Websites


Referenced images

Figure 1.1 - Glow Sticks no 1. - Jackson, Thomas, emergent behavior - http://thomasjacksonphotography.com/emergent-behavior.html

Figure 1.4 - http://hive-open.org/documentation/interactive-dance-floor

Figure 1.5 - http://hive-open.org/documentation/interactive-dance-floor

Figure 1.6 - http://www.designdriveninnovation.com/TE.html

Figure 1.8 - http://duino4projects.com/debounce-an-input-using-arduino/

Figure 1.10 - ATMO, final prototype Interactive Environments minor at the Delft University of Technology - http://studiolab.ide.tudelft.nl/ie/minor-201415/atmo/

Figure 1.12 - courtesy to Chris Kievid

Figure 1.13 - BrightLogic - http://brightlogic.com/main/taxonomy/term/48

Figure 1.14 - http://www.baggagereclaim.co.uk/overcoming-your-fear-of-having-standards-in-your-relationships/

Figure 2.1 - Saro, Yuki - http://cdn.zmescience.com/wp-content/uploads/2015/06/fireflies-Yuki-Saro-3.jpg

Figure 2.4 - Buxton, Bill Sketching User Experiences: Getting the Design Right and the Right Design [Book]. - San Francisco USA : Morgan Kaufmann Publishers Inc, 2007.


Figure 2.6 - Buxton, Bill Sketching User Experiences: Getting the Design Right and the Right Design [Book]. - San Francisco USA : Morgan Kaufmann Publishers Inc, 2007.

Figure 2.11 - Nagesh Javali, G. -  http://www.csc.villanova.edu/~mdamian/Research/CDCF0728909.html


Figure 2.17 - Boltrukiewicz, Joseph - http://www.digitaljournal.com/image/98082

Figure 2.23 - https://en.wikipedia.org/wiki/Cellular_automaton


Figure 3.8 - http://www.heijmans.nl/en/projects/van-gogh-roosegaarde-cycle-path/

Figure 4.15 - Catharinaplein from a Google World's perspective

Figure 6.1 - https://markdeeble.wordpress.com/2014/02/09/here-today-gone-tomorrow/

Figure 7.1 - Paciorek, Dariusz - http://www.wired.com/2013/03/powers-of-swarms/

Figure 8.1 - Marshmallows no 1. - Jackson, Thomas, emergent behavior - http://thomasjacksonphotography.com/emergent-behavior.html

Figure 8.2 - http://7-themes.com/6876385-hot-air-balloon-wallpaper-hd.html
9.1 Design brief

EINDHOVEN AS TESTING GROUND FOR INTELLIGENT LIGHTING

The well-known Dutch performer Freek de Jonge wrote the verse: The city of light is light, not light as in weight, oh no, the reason behind its lightness, is the light bulb and its glow. That is no longer the case. Sales of light bulbs have been discontinued since 2012. The lamps that illuminate our annual commemorative Liberation Day Light Route have been replaced by LED lamps. However, it is no longer about the kind of the lamps in the city. It is about innovative lighting systems for public space. These days, lighting is digital and no longer depends on an underground power source. Lighting is increasingly an integrated part of roads and objects and provides not only light, but also communication and atmosphere. Visitors to the Eindhoven light art festival ‘GLOW’ experience this every year.

In order to assess the possibilities of smart light, we are making various areas of the city available as living laboratories. We are asking the market to continue developing ‘new lighting’ up until 2030, collaborating with us as the government, but also with knowledge institutions and importantly, with the residents of Eindhoven. Together, we are working to create a public space that adapts to the wishes and needs of its users and responds to new possibilities with lighting. In this way we improve, together, the quality of life in our city.

A VISION FOR 2030

Eindhoven will have an omnipotent smart light-grid that combines IT, energy and lighting. This synergy allows for other usages and possible behaviour designs of social spaces.

Initial research has indicated some inspirational scenarios in which technological opportunities allows for the realization of this vision:

- Creativity, e.g. light graffiti;
- Personal lighting, e.g. accompanying tailored travelling light and navigation;
- Multi-media light applications, e.g. functional and artistic;
- Playing with light, e.g. interactive games;
- Interactivity, e.g. meeting points;
- Invisible infrastructures, e.g. self-sufficient systems;
- Safety-features, e.g. braking distance projection;
- Atmosphere creation, e.g. stores and restaurants;
- Shifting boundaries, e.g. experiments GLOW;
- Civilian participation, e.g. implementations in neighborhoods.
The Catharinaplein has been designated as such a living laboratory. The Catharinaplein has endured vacancies, poor hydraulic works, dying trees and decaying tiling. A renovation of the Catharinaplein calls for the development of an invisible infrastructure that invites exploration of the Catharinaplein and its surrounding environment. This infrastructure should invite for social interaction between residents and tourists to in turn draw in new establishments and revive the liveliness of the square.

To accommodate the entire area of the square and quite possible expand before 2030, a distributed network of interactive floor tiles is purchased to fit the modular and scalable characteristics of the brief. While the network of tiles is in place, the municipality of Eindhoven is still evaluating the interaction design that will be deployed. You are asked to come up with as many creative interaction designs for this network of interactive tiles. To help you get started, you have been assigned a tool to explore the possible behaviors of a distributed network. While the interaction design should be possible to deploy on a distributed network, do not limit yourself to the available examples of the tool and try to be as creative as possible.
DESIGN PROCESS SETUP

During the research you will use the tool at hand and generate concepts sequentially in 3 phases. Phase 2 and 3 will be closed with a short interview and questionnaire respectively. Please use the concept templates and the stickers provided to sketch and explain your interaction design concept.

During the research please think out loud. The observant might ask you to speak your mind to understand what you are going through. Please ask questions when you have problems understanding things. For research qualities, the observant may choose to not answer.

Figure 9.183 Structure of the design brief