Towards a lock family platform

ir. T. Wilschut (t.wilschut@tue.nl)

Report Number: CST 2015.099

Final report

Advisors: Dr. ir. L.F.P. Etman
Em. prof. dr. ir. J.E. Rooda
Prof. dr. ir. I.J.B.F. Adan

RWS contact: ir. A. Hijdra (arjan.hijdra@rws.nl)
drs. J.A. Vogel (han.vogel@rws.nl)

Eindhoven University of Technology
Department of Mechanical Engineering

Commissioned by: Rijkswaterstaat - Multi-Water-Werk project

Eindhoven, July 27, 2015

Contents

Chapter 1

Introduction

In the 1930's many 'one of a kind' navigation locks have been built. Lock managers have observed that the great variety in lock designs has a negative impact on lock reliability, availability (RA) and life cycle costs (LCC). The negative impact is primarily due to the need for local specialized knowledge to operate and maintain these locks and the need for many expensive and unique spare parts. Many of these locks will reach their end of life or have to be replaced to keep up with the growing shipping traffic in the coming decades. Therefore, now is the ideal moment to re-think the design of this family of locks.

1.1 Objective

Rijkswaterstaat has started the Multi-Water-Werk (MWW) project that is dedicated to develop a modularization and standardization strategy to increase the uniformity among locks designs. By applying modularization and standardization, MWW aims at decreasing lock LCC and increasing RA. Moreover, MWW aims at reducing uncertainty in construction time and costs (UTC) since large complex construction projects often shown significant budget and schedule overruns.

MWW aims at only standardizing those components which have a significant influence on lock RA, LCC, and/or are related to construction activities which have a significant impact on UTC, such that design and innovation freedom is maintained for contractors. Note that while RA and LCC are lock characteristics, UTC is a lock construction process characteristic.

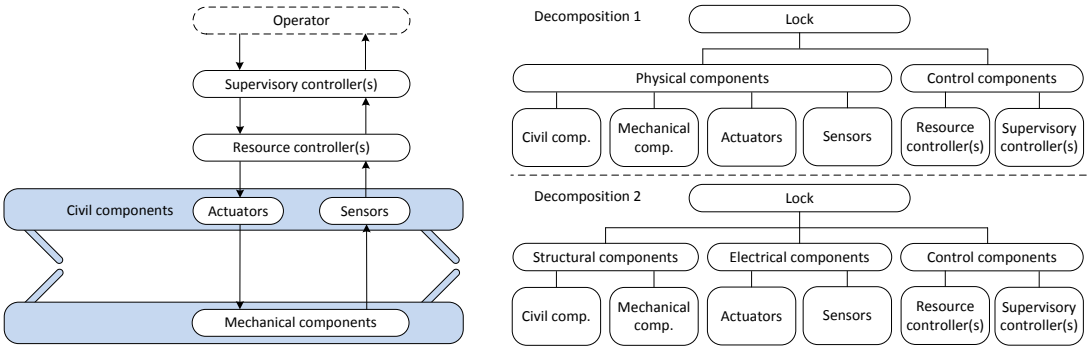
To help develop a modularization and standardization strategy, MWW has hired scientific staff of the TU/e Mechanical Engineering faculty with expertise in systems engineering, life cycle analysis, performance analysis, optimization, and control of systems. In particular, MWW has requested the TU/e to investigate Design Structure Matrix analysis techniques to:

- identify possibilities for modularization and standardization, and to
- identify functions/components/activities which have a significant impact on RA, LCC and/or UTC.

This, in view of realizing a family of fifty-two locks before the year 2040.

1.2 Lock family platform

Locks are produced in relatively low quantities and are usually customized to location specific requirements. Therefore, a significant amount of re-engineering is required before a lock can be



(a) (b)

Figure 1.1: (a) Schematic drawing of a lock system, and (b) two possible lock decomposition trees.

built, i.e. locks are designed and build following an 'Engineer-to-Order' (ETO) approach. Alblas and Wortmann [2014] state that ETO industries can reduce product development time and costs by using function-component product family platforms. Such a platform consists of a 'common function-component core' and optional modules. The common core comprises functions and components which are required in every product. The optional modules comprise functions and components which are not required in every product but can be added to customize a product. Functions and components which are part of the platform may either be fully standardized, partially standardized, or non-standardized, allowing the ETO industries to create a 'nuanced design freeze' of their products.

It is postulated that a function-component lock family platform can help MWW to achieving its goals, i.e. by creating a nuanced lock design freeze MWW can increase lock RA, decrease LCC and UTC while maintaining design freedom for contractors. Moreover, current political policy dictates that public tenders should be written on a functional basis. In practice, RWS writes functional requirements for conceptual lock components, i.e. components of which the physical shape is yet to be determined. As such, a function-component lock family platform is coherent to this political policy and the current activities of RWS.

In this study multiple Design Structure Matrix based analysis techniques are investigated that can be used to identify the common function-component core, optional function-component modules, and functions, components and activities which may be suitable candidates for (partial) standardization. A recent overview of Design Structure Matrix methods and applications is presented in Eppinger and Browning [2012].

1.3 Lock system

Lock modularization is the first step in developing a lock function-component family platform. Figure 1.1a shows a schematic drawing of a lock which consists of civil components, mechanical components, actuators, sensors, resource controller(s), and supervisory controller(s). These components jointly provide the three main lock functions: 'Passeerfunctie', 'Peilbeheerfunctie', and the 'Keerfunctie'. As such, the components dependent on each other.

Modularization aims at decomposing a system into multiple modules, i.e. groups or clusters of components that can be designed and manufactured relatively independently of the rest of the system. Often a decomposition is depicted by means of a tree like picture, as illustrated in Figure 1.1b. The decomposition in the top of Figure 1.1b decomposes the lock into two groups, the modules 'Physical components' and 'Control components'. The decomposition in the bottom of

decomposes the lock intro three groups, the modules 'Structural components', 'Electrical components' and 'Control components'. Both trees are equally valid and both are equally arbitrary. In fact, one could create many other decomposition trees. However, determining which modular decomposition is best, is not a trivial task.

In general, it is desired to decompose a system into modules that are as independent as possible (Suh [1998], Pahl et al. [2007]). To find such modules one has to know which components depend on each other. The DSM provides a compact and analytically advantageous format for modeling and visualizing dependencies between components. In fact, a DSM provides an abstract representation of the lock design. Clustering algorithms can be used to permute the rows and columns of the DSM in order to reveal the structure within the network of dependencies. Based upon the revealed structure it is possible to identify component modules.

One could also model dependencies between lock functions and identify function modules or model dependencies between activities of a lock construction project and identify activity modules. In general, one can model dependency structures on various levels of abstraction, between different types of elements, e.g. components or functions, and one can consider different types of dependencies, e.g. functional, spatial and information dependencies. In this study, it is investigated which dependency structures are relevant with respect to the objectives of MWW, i.e. the modularization and standardization of locks and the identification of functions, components and activities with a significant impact on RA, LCC and UTC. First, the lock life cycle is studied since RA, LCC and UTC are characteristics which relate to different stages of the lock life cycle.

1.4 Lock life cycle

Within the lock life cycle five phases can be distinguished (Renders and Rooda [2004], Nagel and Tomiyama [2004]). First, the orientation phase covers the exploration of needs of a new lock or lock replacement. Secondly, once the needs are clarified, one enters the specification phase, which covers the design stages of the lock, specifically: functional design, conceptual design, embodiment design, and detailed design. Third, the lock is constructed in the realization phase. Fourth, the lock enters the utilization phase, which is approximately one hundred years. Utilization means operation, maintenance, and revision. Fifth, upon the end of its life time, one enters the elimination phase, which either means disposing the lock, or replacing it with an entire new lock. We refer to these various stages in the life cycle as engineering domains.

Each domain may be decomposed in to elements, for instance, sub-functions for the functional domain, components for the conceptual design domain, or operational activities for the operational domain. Typically, there are dependencies between these elements. As such, for each domain it is possible to create a DSM describing the dependency structure for that particular domain. What is more, there are also dependencies between the various domains. Dependencies between two domains can be modeled using a design matrix (DM). By assembling the various DSMs and DMs into a single matrix the life cycle multi-domain matrix is obtained. The DSMs describe the dependencies between elements of a single domain. The DMs describe the dependencies between two domains. The life-cycle multi-domain matrix provides a structured representation of the lock life cycle.

1.5 Lock family

Looking at the lock life cycle matrix, the functional and conceptual design domains are of particular interest for MWW since functional requirements have to be specified for conceptual components for the family of fifty-two locks. By increasing or decreasing the level of detail on which the functional requirements are specified, MWW can increase or decrease the design freedom contractors have.

In this study, a function-component design matrix is created based on data of seven locks. The function-component DM is used to generate a function DSM and a component DSM. A clustering algorithm is applied to find function-component modules. The found modules give an initial indication on how the function-component lock family platform could be organized. Moreover, a ranking algorithm is developed which determines which components yield most loss in lock functionality in case of failure. The high ranking components are of particular interest with respect to lock RA and may be suitable candidates for (partial) standardization.

One may argue to what extend the function-component dependency structures represent the physical dependency structures of a lock, i.e. the dependencies between components within the physical domain, and to what extend the results obtained from analyzing the function-component dependency structures with respect to modularization and standardization are representative for a real lock. Therefore, a case study considering a single existing lock has been conducted.

1.6 Case study lock Eefde

Over a period of several months Dijkstra [2015] has created a DSM describing the energy, information, spatial and location dependencies between fifty components comprising lock Eefde. Dijkstra used a clustering algorithm to reveal to structure of lock Eefde and identify component modules. Additionally, Dijkstra modified the method of Brady [2002] to identify components and interfaces with a significant impact on reliability and availability. The DSM analysis of lock Eefde with respect to RA yielded insightful results regarding lock component dependencies and provides an initial indication on which components may be suitable candidates for standardization.

In line with the work of Dijkstra, the dependency structure of Eefde may also be analyzed with respect to LCC in which a distinction has to be made between initial construction cost, regular annual maintenance cost and periodic renovation costs. We have initiated the collection of data for such an LCC analysis, but upon the end of the contract period, the required data was not yet available to us.

1.7 Lock construction

Looking at the lock life cycle matrix, it is observed that UTC resides in the manufacturing domain. As such, UTC is a characteristic of the lock construction process, not of the lock itself. A construction process can be decomposed into a number of dependent activities, i.e. an activity may build upon the results of a previous activity. However, some activities are not correctly completed at the first attempt, but have to be redone. This phenomenon is called rework and is thought to be a major cause of schedule and cost overruns of construction projects.

In this study we are particular interested in the risk of exceeding construction time and cost (RTC). Risk is not the same as uncertainty, while uncertainty is a potential, unpredictable, unmeasurable and uncontrollable outcome, risk is a consequence of action taken in spite of uncertainty (Antunes and Gonzalez [2015]). Thus, the course of activities during a construction process will influence RTC.

Schuijbroek [2015] and Mommers [2015] have effectively implemented the method of Browning [2001] to simulate the cost and duration of a lock construction process. Using a simplified lock construction process they investigate the effect of rework on costs and duration. Moreover, they use a so called criticality matrix to identify activities which may have a significant impact on RTC.

The simplified lock construction case presents promising results. Extension and application of the DSM method for a more detailed representation of the lock construction process would be a next step to identify modularization and standardization options that reduce the risk of construction time and budget overruns.

1.8 Outline

Chapter 2 discusses several DSM related techniques which are used throughout this study. Chapter 3 explains the ideas and concepts behind the life cycle structure matrix. Chapter 4 investigates the function-component dependency structure of a single lock. Chapter 5 elaborates on Chapter 4 and investigates the function-component dependency structure commonality of a group of seven locks. Chapter 6 introduces the RA-ranking algorithm which detects functions and components which yield a significant loss in functionality in case of failure. Chapter 8 discusses the effect of rework on construction time and costs. Finally, the conclusions and recommendations following this nine month study are given in Chapter 9.

Chapter 2

DM, DSM and MDM

The design structure matrix (DSM) technique has been selected as a means for modeling, visualizing and structuring dependencies in lock systems. In literature one can find many DSM variants and other matrix based techniques for modeling dependencies. In this study a design matrix (DM), a product and a process design structure matrix (DSM), and a multi-domain matrix (MDM) are used. In this chapter the basic concepts behind these four matrix representations are explained.

2.1 Design matrix DM

A design matrix (DM) (Suh [1998]) is a binary $N \times M$ matrix, mapping the dependencies between a set of N row elements and a set of M column elements. The row and column sets should be disjunct, i.e. no elements which are part of N, can be part of M and vice versa. Table 2.1 shows an example DM, where $N = \{a, b, c, d, e\}$ and $M = \{k, l, m, n, o, p\}$; an x mark at position i, j denotes that element i depends on element j.

	k	l	m	n	o	p
a		x			x	x
b	x		x			x
c		x			x	
d			x	x		
e	x		x			

Table 2.1: A design matrix, a mark at position (i, j) indicates that element i depends on element j.

The row and columns sets typically consist of elements from different domains. For example, the row set may consist of system functions while the column set may consist of system components. As such, a DM is often referred to as a domain mapping matrix, defining the relations between two domains.

2.2 Design structure matrix DSM

A DSM (Steward [1981]) is a square $N \times N$ matrix with identical column and row labels. It is also referred to as an N^2 matrix. Table 2.2 shows two DSMs in which the system elements are denoted a, b, c, d, e. Off-diagonal x-marks denote a relationship between two elements. Reading along

a row i reveals which elements, element i depends on. Reading down a column j reveals which elements are dependent on element j. The dependency structure may be unidirectional which yields a symmetric DSM, as illustrated by Table 2.2a, or directed which yields an asymmetric DSM, as illustrated by Table 2.2b.

	a	b	c	d	e
a	-	x			x
b	x	-	x		
c		x	-		x
d				-	
e	x		x		-

(a)

	a	b	c	d	e
a	-	x			
b		-			
c		x	-		x
d				-	x
e	x		x		-

(b)

Table 2.2: An undirected (a), and a directed (b) DSM. A mark at position (i, j) indicates a relation between element i and element j.

Eppinger and Browning [2012] gave an extensive overview of published DSM applications in industry. Based on this study, they defined four DSM types, being: the product DSM, the organization DSM, the process DSM and the multi-domain matrix. Except for the organization DSM, these DSM types will be explained in further detail. In this study for RWS the product DSM, the process DSM and the multi-domain-matrix are explored. These matrix types are explained in further detail below.

2.3 Product DSM

A product DSM describes the dependency structure between components of a system. Components of a system do not change over time, as such the dependency structure is static. Static DSMs are usually analyzed using clustering algorithms. Through clustering one aims to find groups of highly dependent components, i.e. groups of components which have a mutual higher dependency compared to components which belong to other groups. The grouping that results from clustering indicates natural lines along which a system can be decomposed into modules.

For example, Table 2.3a shows a product DSM in which no clusters are clearly visible. However, after application of a clustering algorithm, permuting the rows and columns of the DSM, one obtains the DSM shown in Table 2.3b. In the clustered DSM it can be clearly seen that the system consists of two independent component clusters. Three popular clustering algorithms are hierarchical clustering (Schaeffer [2007]), k-means clustering (Fortunato [2010]) and Markov clustering (Van Dongen [2008]).

	a	b	c	d	e	f
a	-		x		x	
b		-		x		x
c	x		-		x	
d		x		-		x
e	x		x		-	
f		x		x		-

(a)

	d	b	f	e	a	c
d	-	x	x			
b	x	-	x			
f	x	x	-			
e				-	x	x
a				x	-	x
c				x	x	-

(b)

Table 2.3: An example product DSM before (a) and after clustering (b).

2.4 Process DSM

A process DSM captures the dependencies between activities of a process. The arrangement of activities along the axis of the DSM indicates in which sequence the activities are carried out(from top to bottom). Lower diagonal DSM entries denote feedforward marks, i.e. entry (i, k) with $k < i$ denotes that activity i depends on upstream activity k which is carried out before i. Upper-diagonal DSM entries denote feedback marks, i.e. entry (i, k) with $k > i$ denotes that activity i depends on downstream activity k which is carried out after i (Eppinger and Browning [2012]).

In general, feedback marks are undesired since they may induce rework. That is, when starting activity i, one has to estimate the result of downstream activity k which is not yet completed. After completion of activity k, one may find a difference between the estimate and actual result of activity k. As such, it may be necessary to redo activity i since it may have been performed based on faulty input information. This effect is called rework, and may cause an increase of the total process completion time.

Since activities usually relate to time, process DSM are often referred to as time-based DSMs. Process (or time based) DSMs are typically analyzed using sequencing algorithms. Through sequencing one aims at minimizing the number of feedback marks (Meier et al. [2007]). Popular sequencing methods are genetic algorithms (Reeves [1995]), simulated annealing (Brooks and Morgan [1995]), tabu search (Glover [1989, 1990]) and ant colony optimization (Dorigo et al. [2006]).

The effect of a sequencing algorithm is illustrated by the process DSMs shown in Table 2.4. The process DSM Table 2.4a has four feedback marks. However, changing the activity sequence from a, b, c, d, e, f to e, b, d, c, f, a removes all feedback marks as seen in Table 2.4b. As such, all activities can be executed sequentially lowering the chance of rework.

	a	b	c	d	e	f
a	-		x			x
b		-			x	
c			-		x	
d		x		-		
e					-	
f		x		x	x	-

(a)

	e	b	d	c	f	a
e	-					
b	x	-				
d		x	-			
c	x			-		
f			x	x	-	
a				x	x	-

(b)

Table 2.4: An example DSM before (a) and after sequencing (b).

2.5 Multi-domain matrix MDM

A multi-domain matrix (MDM) is a block diagonal matrix of which the diagonal consists of single domain DSMs. The off-diagonal blocks are called domain mapping matrices (DMM) and relate elements of one domain to elements of another domain. Figure 2.1 shows a schematic MDM in which a function DSM is related to a component DSM via a function-component DM. The function DSM captures the dependencies between functions, the component DM captures the dependencies between components, and the function-component DM captures the dependencies between functions and components.

In Figure 2.1 the MDM denotes the relations between the function and component domains, which are both product DSMs. However, one could also link product DSMs to process DSMs. For example, one could relate components to activities in a manufacturing process.

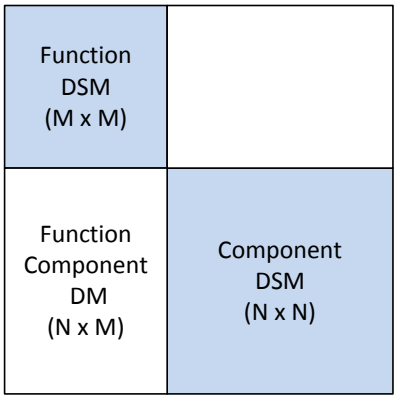


Figure 2.1: Schematic multi-domain matrix (MDM).

Chapter 3

Lock life cycle MDM

This chapter presents a life cycle multi-domain matrix (LCMDM). The purpose of this matrix is to visualize the various phases and elements that are of importance during the life cycle of a lock, especially with respect to reliability and availability, life cycle cost, and risk of exceeding planned building time and cost. The LCMDM is used to illustrate how the functional requirements, stated by RWS, relate to these performance characteristics.

3.1 Life cycle phases

The LC of a lock and its accompanying manufacturing process, is divided into 5 phases (Renders and Rooda [2004], Nagel and Tomiyama [2004]):

1. **Orientation** covers the period of time in which the needs (N) and requirements are explored, from the customer perspective.
2. **Specification** covers the period of time in which the customer needs are translated into a appropriate lock design. This translation process comprises four systems engineering domains (Suh [1998], Pahl et al. [2007]):
   - 2.1 **Functional design domain** (F) entails the definition of functions and their dependencies which are required to full-fill the customer needs.
   - 2.2 **Conceptual design domain** (C) denotes to the selection of technologies which are able to deliver the required functions.
   - 2.3 **Embodiment design domain** (E) is the exploration and evaluation of several design alternatives.
   - 2.4 **Detailed design domain** (D) is the final step of the specification phase in which the most preferable design is engineered in full detail.
3. **Realization** denotes the actual physical realization of the system, i.e. the manufacturing (M) of the system.
4. **Utilization** is the period of time in which the system is utilized. Within this phase three dominant domains are identified:
   - 4.1 **Physical domain** (P) describes the lock 'as built'. In theory, P is a 'one on one' physical realization of D.

4.2 **Operational domain** (O) concerns the manner in which the system is operated and maintained. For instance, the work-cycle is part of this domain.

4.3 **Revision domain** (R) covers the periodic review of the operational performance and physical status of the system. This revision may induce large-scale maintenance such as the replacement of a control system.

5. **Elimination**: follows when the system has reached its end of life or is no longer needed. In this phase the system is demolished (annihilation (A)).

The LC's of a product and its manufacturing system are not independent. That is, the manufacturing system is needed to realize the product. Therefore, the design of the manufacturing system is highly dependent on the design of the to be produced product, and vice versa. This implies for the lock system that to control risk of exceeding planned realization time and cost it is important to have a thorough understanding of the product, the manufacturing system, and the interplay between these two systems.

In principle any of the identified domains can be viewed from the static or dynamic (time-based) perspective. For example, in the conceptual design domain one can regard the dependencies between conceptual components describing the conceptual product (static) or the dependencies between design activities (dynamic) describing the design process. What is most interesting, depends on one's objectives.

3.2 Life cycle multi-domain matrix LCMDM

The five life cycle phases are characterized by one or more dominant domains of interest. Each domain may be decomposed into multiple dependent elements. For example, the N domain can be decomposed into requirements. These requirements are usually dependent, e.g. an energy-consumption and a power requirement need to be non-conflicting. These dependencies can be modeled using a design structure matrix (DSM).

Apart from these inter-domain dependencies, there are intra-domain dependencies, i.e. the requirements stated in the N domain influence the functions in the F domain. These dependencies can be modeled using a domain mapping matrix (DMM). Combing the domain DSMs and the DMM's into a single multi-domain matrix (MDM), yields the life cycle multi-domain matrix (LCMDM) depicted in Figure 3.1.

At the bottom of the LCMDM figure one can find the five life cycle phases. Reading from the top left down to the bottom right one can find the dominant domains. These domains are sequentially coupled via lower-diagonal DMM's. Showing the flow of information from one domain to the next (feedforward). In practice upper-diagonal DMMs are also present (feedback), i.e. engineers usually have to iterate several times between domains before a suitable design is obtained.

Of particular interest, with respect to the LCC objective, is the N $\times$ R upper-diagonal DMM. Periodic review will yield new needs and requirements. For example, environmental legislation may have been changed. As a result, one has to (partially) reconsider the design of the system. These revision cycles significantly contribute to the LCC. As such, it is beneficial to use flexible lock designs which can easily evolve over time (Keese et al. [2007], Cardin et al. [2013]).

RTC is a performance characteristic of the manufacturing domain (M). RA are a result of the interplay between the elements of the physical (P) domain and the elements of the operational (O) domain. However, political policy dictates that the main activities of RWS should be focused on the needs (N), functional (F), operational (O) and revision (R) domains. The main portion of design and manufacturing is carried out by third party contractors. Public tenders to hire the contractors are primarily made on a functional basis.

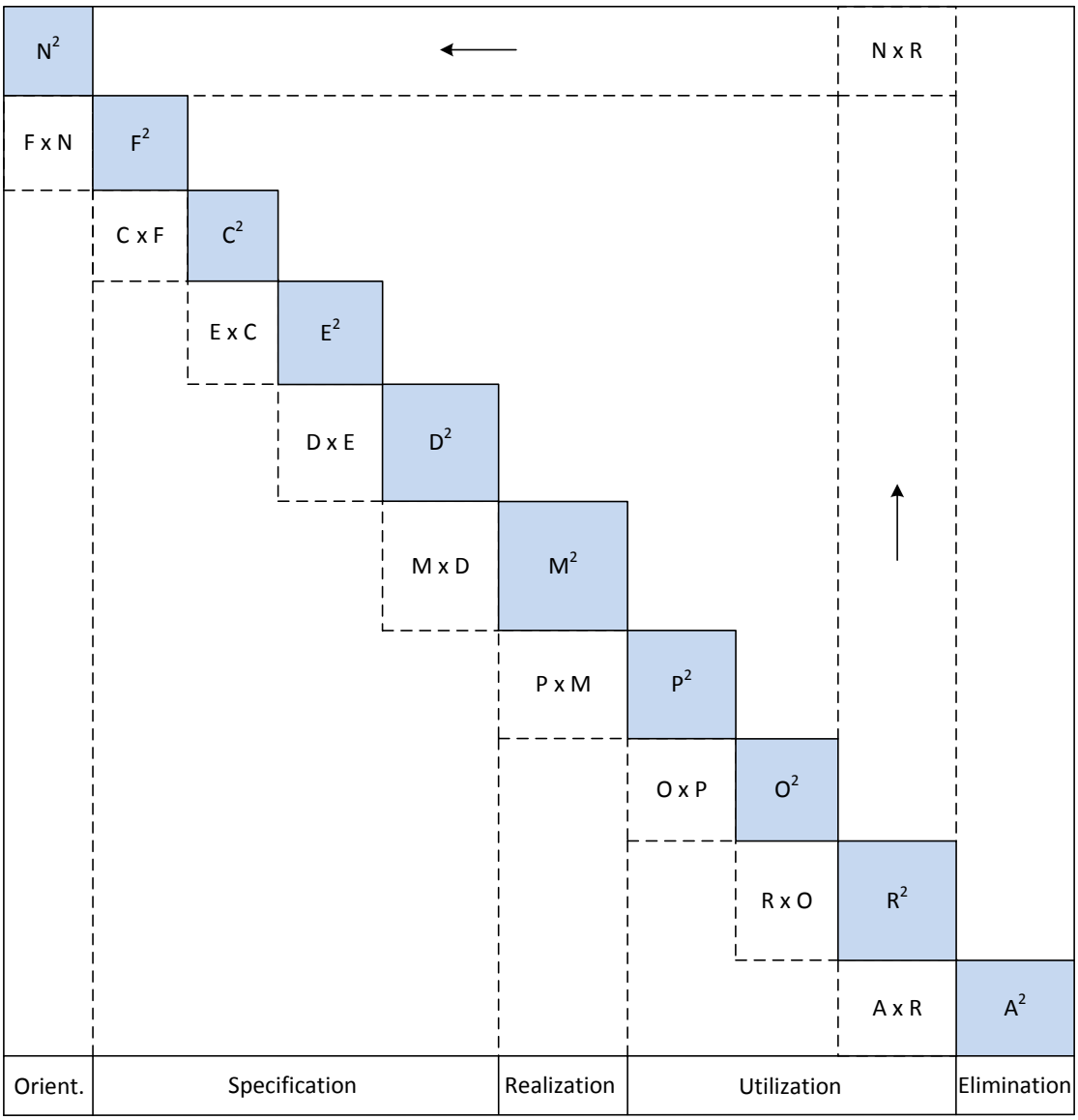


Figure 3.1: The life cycle multi-domain matrix

The contractors are responsible for the specification phase and the realization phase, during which the functional tender is converted into a physical lock (P). As part of this process, the contractors have to make thousands of decisions. All these decisions may influence the RTC, RA and LCC for which RWS is held accountable. However, RWS has little influence on the choices made by the contractors. Moreover, the incentives the contractors have in making these decisions may not be in line with the goals of RWS. As such, it is very difficult, if not impossible, to achieve the goals of MWW from the main RWS domains only (N, F, C, O, R).

3.3 Family LCMDM

MWW concerns the realization of 52 locks. The life cycle of each lock can be described by a LCMDM as depicted in Figure 3.2. Locks have a high degree of similarity, i.e. they roughly consist of the same components. As a result, it is possible to apply standardization and modularization which allows for reuse of components/modules in multiple locks.

Standardization and modularization have multiple advantages. First of all, it leverages development cost over multiple locks, i.e. a component/module only has to be designed once and can be used multiple times. Moreover, during revision, a component/module has to be redesigned once and can again be installed in multiple locks. Effectively, the LCC of the lock family can be reduced. Secondly, it limits the number of choices contractors can make during design and manufacturing. This reduces RTC and increases the influence of RWS on the physical form of the lock (P). Third of all, since the physical form of the lock is more predictable, RWS can make standardized operational and maintenance procedures which apply to multiple locks. Moreover, the number of spare parts can be reduced. This may increase RA, and further decrease LCC.

A disadvantage of standardization and modularization is that it limits the possibilities for innovation by contractors and for the system to evolve over time. Therefore, it is proposed to create a function-component lock platform which allows for a nuanced design freeze while still providing possibilities for innovation, and more importantly increase the influence of RWS on RA, LCC and RTC. Section 5.1 will discuss this platform in more detail.

3.4 Conclusion

The lock life cycle consists of five phases which are characterized by a total of ten domains. The life cycle multi domain matrix (LCMDM) shown in Figure 3.1 is suited for the description of the lock life cycle and shows the dependencies between the life cycle phases and domains.

Lock reliability and availability (RA) is a result of the interplay between the physical and operational domains. Risk of exceeding building time and costs (RTC) resides in the manufacturing (M) domain. Life cycle cost (LCC) are build up during the complete life cycle of the lock.

The main focus of the RWS policy is on needs (N), functional design (F), conceptual design (C), operational (O), and revision (R) domains. Third part contractors make thousands of decisions within the embodiment design (E), detailed design (D) and manufacturing (M) domains which may all influence RA, RTC and LCC. As such, it is very difficult, if not impossible, for RWS to control RA, RTC and LCC.

Modularization and standardization can reduce the number of choices third party contractors can make, can leverage development and revision cost over multiple locks, can reduce the number of required spare parts and allows for standardization operational and maintenance procedures. All of which may positively influence RA, RTC and LCC.

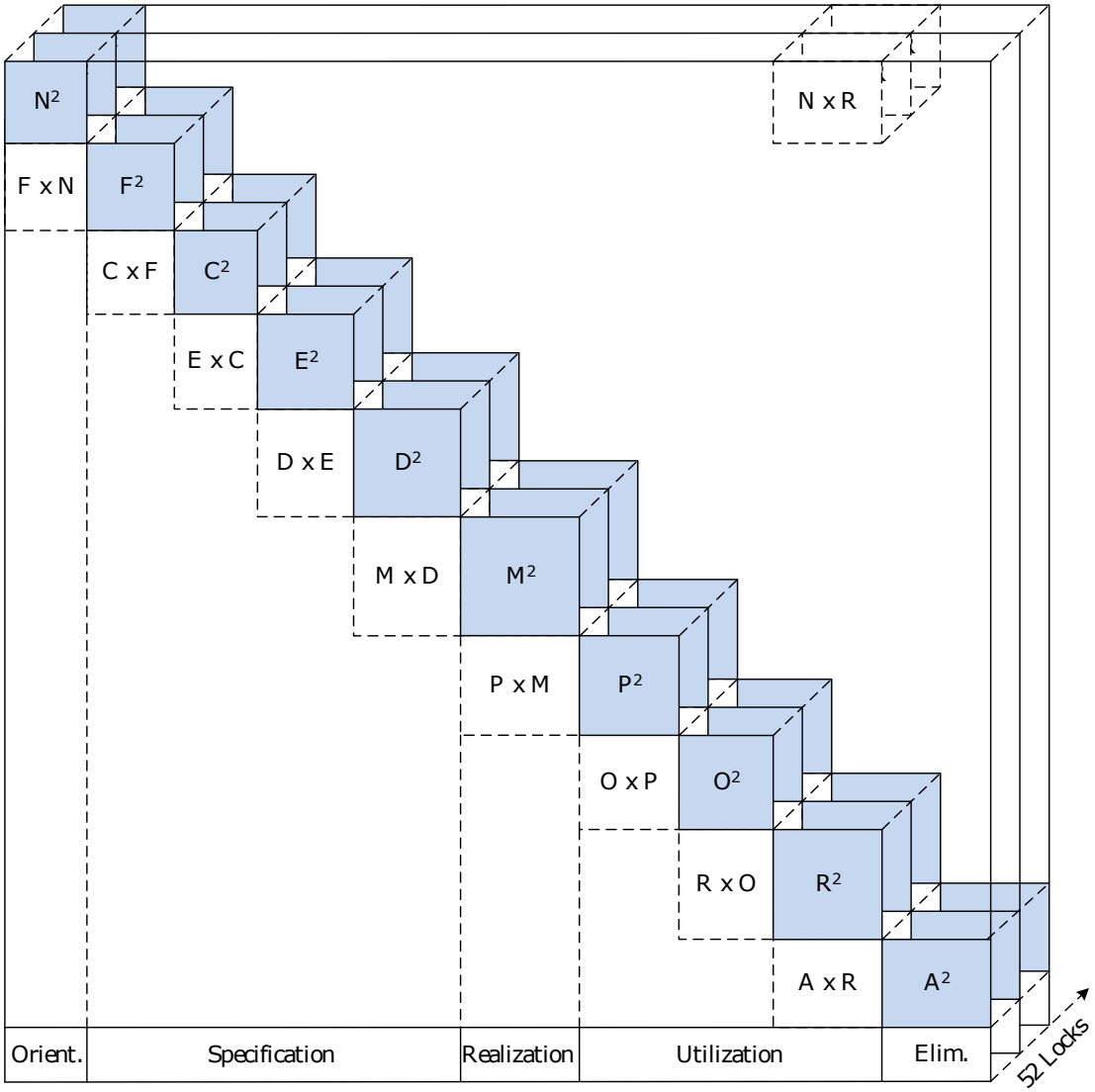


Figure 3.2: The family life cycle multi-domain matrix

Chapter 4

Lock system modularization

Standardization and modularization could significantly contribute to the goals of MWW. This chapter discusses a method for the identification of suitable locks modules. The method is applied to the functional (F) and conceptual (C) design domains. The reasoning behind the selection of these domains is twofold. First of all, both F en C are within the main focus of RWS policy. Secondly, the 37 existing locks have a high degree of similarity on the F and C design levels. Therefore, the method presented in this chapter is also suitable for the modeling and analysis of the lock family structure, which is discussed in Chapter 5

4.1 Function-component DM

Recently, RWS has started with the development of the GRIP environment using the Relatics software (Relatics [2003]). This environment is used to link needs (N), to conceptual components (C), and to link conceptual components to functions (F). Based upon this data it is possible to create N $\times$ C and C $\times$ F design matrices (DMs). In this study the main focus is on the C $\times$ F DM, since these domains are within the main focus of RWS policy.

Figure 4.1 shows an example C $\times$ F DM. The rows are labeled with the names of six conceptual components, the columns are labeled with the names of five functions. A blue mark at position i, j indicates that component i has to contribute to the fulfillment of function j. In fact, each mark in de DM indicates a design decision, i.e. a choice to let component i contribute to function j.

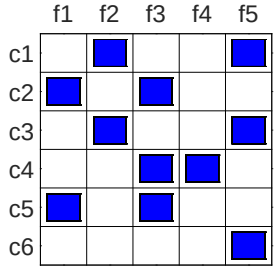
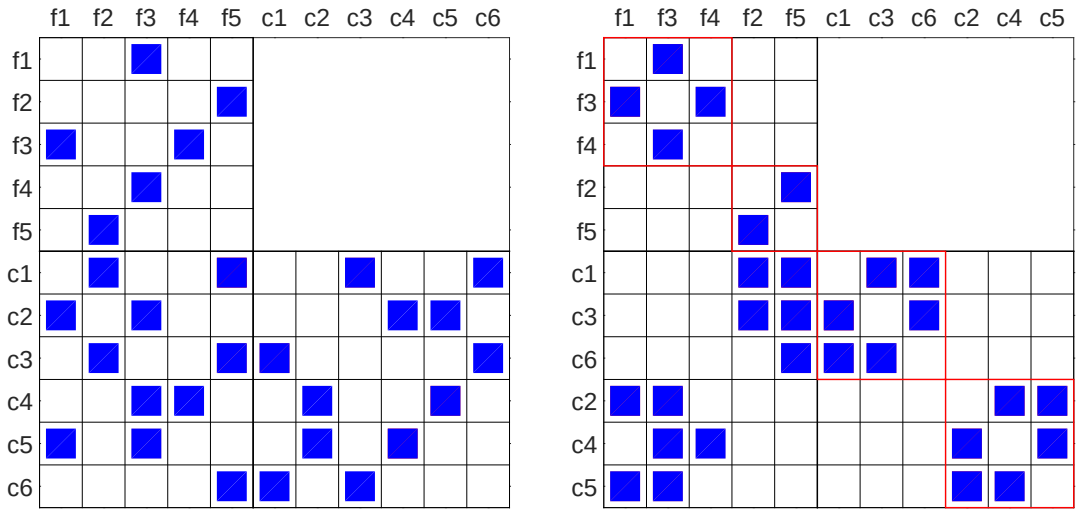


Figure 4.1: A C$\times$F component-function design matrix. A mark at position i, j indicates that component i has to contribute to fulfillment of function j.



(a) (b)

Figure 4.2: A function-component multi-domain matrix: unclustered (a) and clustered (b).

4.2 DM to DSM conversion

In Figure 4.1, components c2 and c5 both contribute to the fulfillment of functions f1 and f3. As such, there is a dependency between c2 en c5. Similarly, functions f3 and f4 are fulfilled by the same components c4. Implying a dependency between f3 and f4, i.e. if one modifies c4 to change f3, f4 may be affected as well.

The marks in the DM determine the dependency structure between components as well as the dependency structure between functions. Following this line of reasoning, it is possible to generate a function DSM (FDSM) and a component DSM (CDSM) from the DM (Maurer [2007]).

Figure 4.2a gathers three matrices. The lower left matrix is the DM of Figure 4.1. The upper left matrix contains the FDSM that has been generated following the aforementioned reasonings. Thus, a mark at position i, j in the FDSM indicates that functions i and j have at least one component in common which contributes to their fulfillment. The lower right matrix is the CDSM that has been generated from the DM. As such, a mark at position i, j in the CDSM indicates that components i and j have at least one function in common to which they both contribute.

By permuting the rows and columns of both the FDSM and CDSM separately, using a clustering algorithm, it is possible to highlight the underlying system structure. The columns of the DM are permuted following the permutation sequence of the FDSM, the rows of the DM are permuted following the permutation sequence of the CDSM. For example, in Figure 4.2b one can easily identify two functions as well as two component clusters. In the permuted DM it can be seen that components {c2, c4, c5} provide functions {f1, f3, f4}, and that components {c1, c3, c6} provide functions {f2, f5}. As such, it is possible to divide this system into two functionally independent modules. It is important to keep in mind that these modules are not necessarily physically independent despite the fact that they are functionally independent.

4.3 Results

Figure 4.3 shows a MDM, obtained from a GRIP file and clustered with a Markov clustering algorithm (Van Dongen [2008]), in which one can distinguish five function clusters (modules) and eight component clusters (modules). Function cluster 1 consists of rows/columns 2-10, cluster 2 consists of rows/columns 11-14, cluster 3 of consists of rows/columns 15-18, cluster 4 consists of rows/columns 20-28, and cluster 5 contains rows/columns 31-31.

Component cluster 1 consists of rows/columns 3-6, cluster 2 consists of rows/columns 8-12, cluster 3 of consists of rows/columns 16-38, cluster 4 consists of rows/columns 41-42, cluster 5 contains rows/columns 43-51, cluster 6 contains rows/columns 54-56, cluster 7 contains rows/columns 63-71, and cluster 8 contains rows/columns 71-72. The function and component clusters are linked via the DM, for example component cluster 3 provides the functions in function cluster 3.

Looking at the function clusters it is remarkable that the functions 'Bedienen Sluis' (2), 'Noodbedienen sluis' (17), 'Bedienen Sluis (Regulier)' (20), and 'Bedienen en besturen'(21) do not all reside in the same cluster since they are all control functions. Moreover, function 2 ('Bedienen Sluis') is a sub-function of 21 ('Bedienen en besturen'). Therefore, looking at the DM (left lower matrix) one would expect that function 2 is fulfilled by a subset of the components that fulfill function 21. However, this is not seen in the DM, leading to doubts on the correctness and completeness of the grip file.

Additionally, the function 16 'Borgen veiligheid' has a dependency with almost all other functions. This is a result of the fact that a large number of components contribute to this function as seen in column 16 of the DM. It might be the case that this GRIP file was created with focus on safety considerations.

What is more, function 16 'Borgen veiligheid' is a fairly general function, whilst the components which are linked to it are formulated on a much lower abstraction level. This inconsistency in abstraction level yields many component dependencies which do not exist. For example, component 24 'Bliksembeveiliging' has no functional relation with component 27 'Inbraakbeveiliging'. Though they are both related to function 16 'Borgen veiligheid'. Therefore, when building a DM it is desired that both the functions and components are formulated on a similar level of abstraction (Suh [1998]) in order to obtain a useful model of the dependency structure.

Moreover, it is observed that many functions have only been related to components which directly fulfill that function and not to other components which are also required to fulfill that function. For example, there is no dependency between function 21 'Bedienen en Besturen' and component 41 'Energievoorziening'. Therefore, many dependencies in the FDSM and CDSM of Figure 4.3 may be missing.

4.4 Conclusion

The presented method to generate a function DSM and a component DSM from a function-component DM seems an appropriate way to obtain the lock dependency structure. However, when using this method it is required that the functions and components are formulated at a similar level of abstraction and that the functions are related to all components which are needed to fulfill that function.

The DM shown in Figure 4.3 does not meet these requirements. The DM is generated from GRIP data in which functions are related to components. However, the functions and components are not formulated at a similar level of abstraction which yields many component dependencies which do not exist. For example, component 24 'Bliksembeveiliging' has no functional relation with component 27 'Inbraakbeveiliging'. Though they are both related to function 16 'Borgen veiligheid'.

Moreover, functions are only related to components which directly contribute to them with the GRIP files, not to components which indirectly contribute. For example, their is no dependency between function 21 'Bedienen en besturen' and component 41 'Energievoorziening'. As such, many dependencies may be missing in the generated FDSM and CDSM.

Finally, this GRIP data does not contain all lock functions and components. As such, it is not possible to define a suitable lock decomposition based on the GRIP data. Though, if the leveling issues in this GRIP are resolved it could provide a valuable data source for this study.

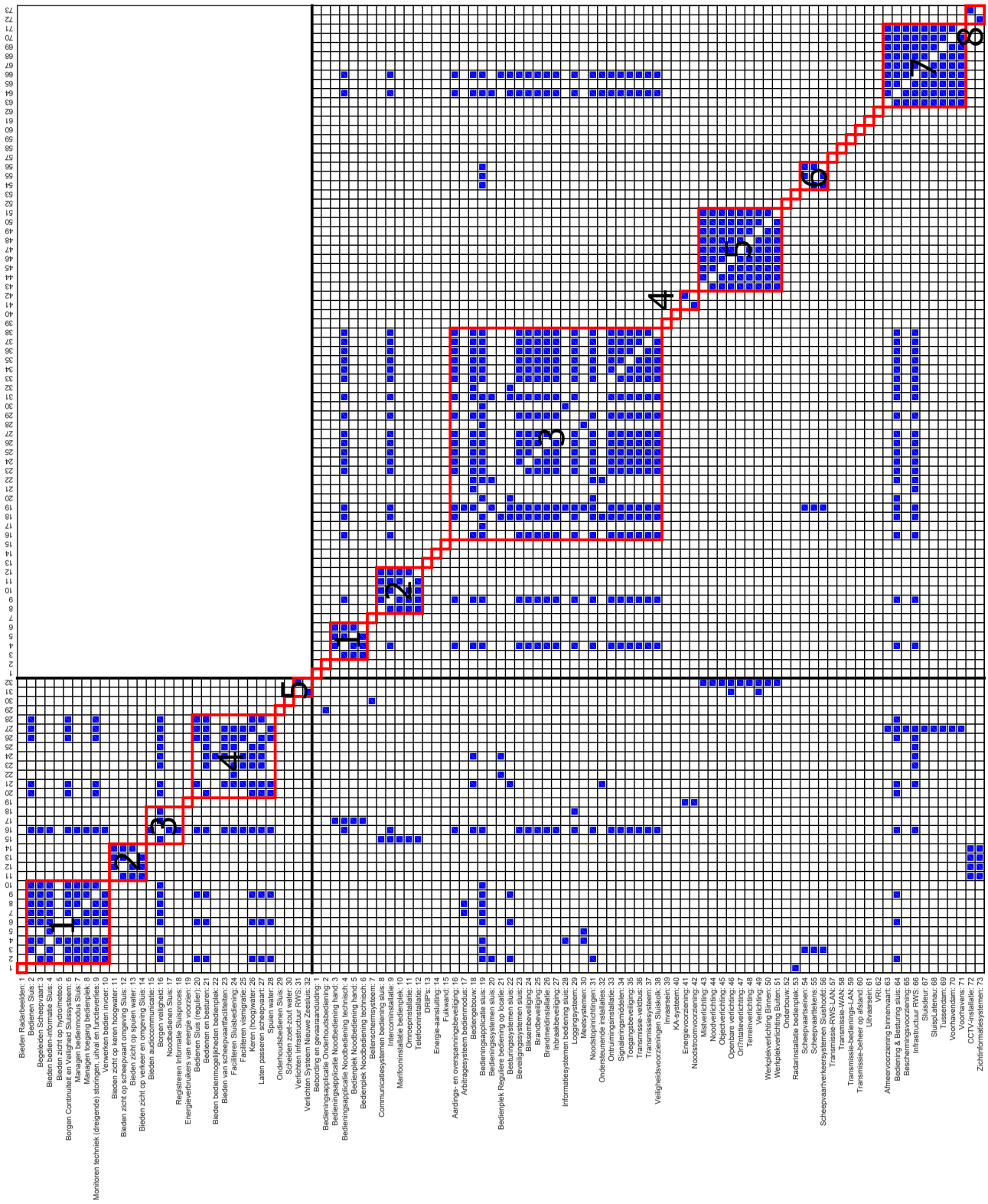


Figure 4.3: MDM based on GRIP data

Chapter 5

Lock family modularization

In literature, standardization and modularization are often associated with product platform design (Jiao et al. [2006]). This chapter focuses on explaining how and in what form such a platform could be used to achieve the goals of MWW, while maintaining design freedom for contractors.

Moreover, a method is presented which can be used in designing such a platform. The method is suitable to model and analyze the function-component dependency structure of multiple locks simultaneously in order to identify a 'common core' and optional lock modules.

5.1 Product platforms

In industry, product platforms are a well know means to achieve economies of scale while accommodating for the increasing demand for product variety across different market niches (Jiao et al. [2007]). A platform usually consists of a common core, i.e. elements which are applicable to all product varieties, and optional modules, i.e. groups of elements which can be added to customize a product. Product platforms appear in three varieties (Alblas et al. [2009]):

- Component centered platforms in which the core and optional modules consist of real physical components which are assembled via standardized interfaces. Such platforms are usually used in 'assembly-to-order' industries such as the automotive branch.
- Function centered platforms in which the core and optional modules consist of system functions and technologies which can provide that functionality. The exact physical shape of those modules is not defined. A function platform can be used to quickly generate conceptual designs. Therefore, function based platforms are primarily used in 'engineer-to-order' industries such as the semi-conductor industry. RWS already has several function centered platform like tools in use, e.g. the water modeling tools SOBEK and SIMONA,the traffic modeling tool BIVAS, and the sluice modeling tool SIVAK.
- Process centered platforms in which the core and modules consists of activities. For example manufacturing steps in a production line. Process platforms are used to create flexible manufacturing systems.

A function centered platform is used in the functional (F) and conceptual (C) design domains, whereas a component centered platform resorts to the detailed (D) design domain, and a process platform belongs to the manufacturing (M) domain. Since F and C are the area of influence of RWS, a function platform would neatly fit within current RWS policy.

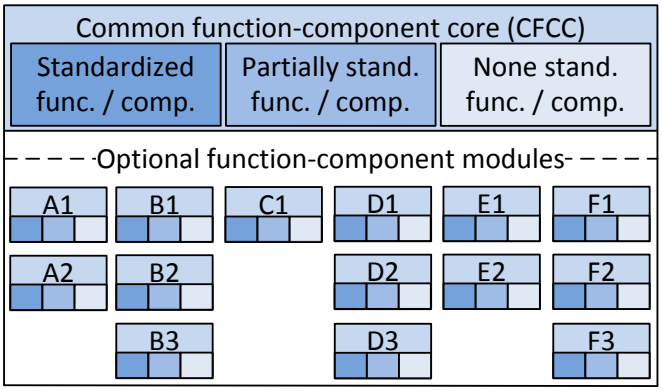


Figure 5.1: Function-component centered platform.

However, to control reliability and availability (RA), risk of exceeding building time and costs (RTC) and life cycle costs (LCC) a component based platform would be better suited. Though, such a platform would leave no room for innovation by contractors. Therefore, it is proposed to seek for a suitable mix of functions and components in a function-component centered platform, as depicted in Figure 5.1. The platform consists of a 'common core' of components and functions which are present in every lock and optional modules containing functions and components which are not present in every lock. The functions/components within the common core and the optional modules are categorized into three groups: standardized functions/components, partially standardized functions/components, and non standardized functions/components.

The standardized functions and components group may consist of those functions and components which have a high impact on RA, LCC, RTC, have a low innovation rate, and a low dependency on situation specific factors. RWS may fully describe standardized functions and may provide preferred or allowed designs for standardized components which fully describe a component and its interfaces. Contractors may change these designs. However, if they do so, they should provide detailed documentation and argumentation on why they wish to change the design and what the impact of these changes is on RA, LCC, and RTC. If these changes have a positive effect, RWS may accept or allow and adopt the new design and use it as the preferred or an allowed design for future component generations.

The partially standardized components group may consist of those functions/components which have a high impact on RA, LCC RTC, have a high innovation rate, and/or strong dependency on situation specific factors. RWS may provide a preferred interface design, i.e. the design of the connections with other components of the system. Partially standardized components are internally free for design by the contractors. Though, the resulting design should be carefully evaluated by the system integrator or RWS. Similar to the standardized components, the standardized interface may be modified by the contractors if it benefits the goals of RWS. Again, RWS may accept or allow and adopt the new design and use it as the preferred or an allowed interface design for future component generations.

The non standardized component group may consist of those functions/components which have little impact on RA, LCC and RTC. These components are free for design by the contractors. However, the contractors have to bear in mind that the non standardized components have to be integrated with the standardized and partially standardized components.

It is important that the process of designing and manufacturing the different standardized, partially and non standardized components is coordinated by a system integrator to ensure the integrability of the lock. The system integration may be coordinated by a department of RWS or outsourced to an engineering agency with experience in system engineering.

By allowing the contractors to change the designs of the standardized and partially standardized components and their interfaces, the platform can evolve over time and include new technology. Moreover, since design changes are only accepted if they have a positive effect on the goals of RWS, the quality of the platform and its designs may improve over time. Furthermore, it is always possible for RWS to modify the preferred designs themselves based on prior experience, preventing design flaws from being made again and ensuring the platform evolves along the desired lock technology road map (Phaal [2009]).

The common function-component core consists of a set of functions and a set of components which are present in every member of the lock family. This common core may be extended with function-component modules. These modules contain sets of functions and components which are not present in every lock but are optional. Each module again consists of standardized, partially standardized, and non standardized functions/components. Moreover, some functions/components within a module could be optional as well. Some modules may only contain standardized functions/components, whilst others may only contain non standardized functions/components. The contractors may also be allowed to change the design of the standardized and partially standardized components within these modules if they provide proper motivation and argumentation. Additionally, it is possible to have multiple versions of the same module. For example, consider two versions of a power supply unit, one providing low voltage power and one providing high voltage power, i.e. the functions of these modules differs.

Such a platform can reduce the number of design decisions a contractor can freely make, can provide room for innovation, and enables reuse of experience obtained in previous projects. Moreover, each design change made to a component or module which has a significant impact on RA, LCC and RTC has to be properly motivated by the contractors and verified by RWS or the system integrator before it is accepted or allowed. Allowing RWS to only accept or allow design changes which contribute to their goals. The platform gives RWS the desired means to gain more grip on RA, LCC, and RTC, while respecting the role of the third party contractors.

5.2 ΣMDM and VMDM

The first step in designing a function component platform is to determine which functions and components belong to the common core and which functions and components belong to optional modules. One can do this by searching for commonality in dependency structure of the lock family members.

In Chapter 4, a method has been introduced which allows for the identification of function-component modules within the structure of a single lock with use of a multi-domain matrix (MDM). Such a MDM can be made for n lock family members. By taking the sum of the n MDMs, as is illustrated by Figure 5.2, one obtains a ΣMDM (Gorbea et al. [2008]). The ΣMDM indicates the frequency of occurrence of function/component dependencies within the n locks. Dependencies with a sum of n are always present and should therefore belong to the common core.

For the ΣMDM approach to work, it is required that the MDM of every lock is complete and correct. If this is not the case, one can resort to a variety MDM (VMDM) (**?**). Similar to the ΣMDM , the rows and columns of the VMDM are labeled with all unique functions and components which are present within the n lock family members. If there is at least one dependency, in at least one lock, between component i and function j a mark is placed in the DM at position i, j. Subsequently, lock experts could directly rate each dependency being common, semi-common or unique. Common dependencies are present in every lock, semi-common dependencies are presents in a few but not all locks, and unique dependencies are present in only one lock.

Experts could also rate the functions and components as being optional or always present. Based on this information it is possible to determine if dependencies are optional or are common.

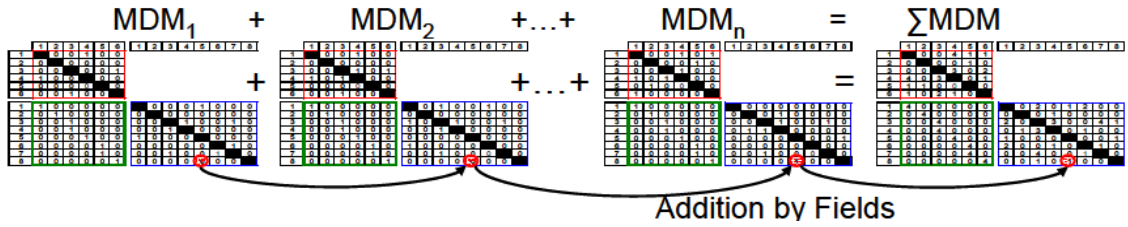


Figure 5.2: The Σ multi-domain matrix (ΣMDM).

5.3 Results

During this study both a ΣMDM and VMDM are created based on existing data. The ΣMDM, depicted in Figure 5.3, is the result of the summation of seven MDMs. These MDMs are generated from FME(C)A files found in the 'RINK dossier'. The data is cleaned and filtered components and functions are formulated on a similar level of abstraction. The GRIP database is still under development. As such, at the moment of research it did not yet contain complete and distinct data on multiple locks. Therefore, the GRIP data is not used in this chapter.

Based upon a FME(C)A file it is possible to create a DM and generate the FDSM and CDSM following the method discussed in Chapter 4. Taking the sum of the resulting MDMs and subsequently clustering both the FDSM and CDSM, yields Figure 5.3. The color of a dependency mark denotes the number of times it was observed, i.e. black denotes it was observed seven times, white indicates it was observed once.

The ΣMDM clearly shows that there are distinct function-component modules present within this group of seven locks. The first module, starting at the top left corner of the CDSM, consists of communication systems, the second of energy supply units, and the third of components which enable a ship to pass through the lock, and the fourth cluster consists of civil components.

Interestingly, none of the component dependencies in the CDSM, as well as none of the function dependencies in the FDSM are observed seven times. Indicating that there would be no common dependencies. This is an unexpected result and is probably caused by the fact that the FME(C)A files were made with a different purpose. Not all FME(C)A files cover all lock components, some only cover a subset. As such, function-component dependencies might not be found within a FME(C)A file despite the fact that they are present.

In the function and object decompositions ('Bomenschutsluis'), which serve as a basis for the 'Object Type Library' (OTL) currently in development at RWS, one can find whether a component/function is optional or common. The black circles along the diagonal in the VMDM shown in Figure 5.4 indicate that a function/component is common, white circle indicate that a function/-component is optional. This information is used to classify the dependencies, shown in Figure 5.3, into three groups and create a variety matrix which is shown in Figure 5.4. A black dependency indicates that both elements are common, a gray dependency indicates that one of the elements is common whilst the other is optional, and a white dependency indicates that both elements are optional.

In Figure 5.4 it can be observed that there are more black dependencies within the FDSM than there are in the CDSM. This indicating that the locks are more similar regarding the functional description then the component specifications. As such, it can be concluded that the diversification of lock designs already starts very early in the lock life cycle. Moreover, it also indicates that engineers have to make a large number of choices during conceptual design which will all influence RA, RTC and LCC.

Looking at the CDSM it can be observed that the module containing the civil components is (not surprisingly) always present. The power supply module always contains 'laagspanningsinstallatie'

and a ‘noodstroomvoorziening’, while the ‘hoogspanningsinstallatie’ is optional. The ship passage module always contains a ‘Bediening- and besturingssysteem’, a ‘Electro- Mechanische uitrusting sluishoofd’, a ’niveaumeetinstallatie’, and a ‘nivelleersysteem. The other components are optional. The communication system modules always contains the ‘Akoestische and Visuele signalering’ and a ‘marifoon’. The components and functions which are always present should belong to the common lock core.

These results give a first indication on how a function-component lock platform could be constructed. However, the functions and components which are considered in Figure 5.3 and Figure 5.4 are only a fraction of the total number of functions and components found in the function and object decompositions (‘Bomenschutsluis’). Therefore, to obtain a suitable platform structure it is required to create and analyze a VMDM containing all functions and components found in ‘Bomenschutsluis’. However, so far no data has been found which relates all functions to all components.

5.4 Conclusion

A function-component centered lock platform can reduce the number of design decisions a contractor can freely make, can provide room for innovation, and enables for reuse of experience and knowledge obtained in previous projects. Moreover, each design change made to a standardized or partially standardized function, component or module which has to be properly motivated by the contractors and verified by RWS before it is accepted. Allowing RWS to only accept design changes which contribute to their goals. The platform gives RWS the desired means to gain more grip on RA, LCC, and RTC, while respecting the role of the third party contractors.

The variety DSM (VDSM) shown in Figure 5.4 gives a first indication on how a function-component lock platform could be structured. However, the functions and components which are considered in this figure are only a fraction of the total number of functions and components found in the function and object decompositions (‘Bomenschutsluis’). Therefore, to obtain a suitable platform structure it is required to create and analyze a VMDM containing all functions and components found in ‘Bomenschutsluis’. However, so far no data has been found which relates all functions to all components.

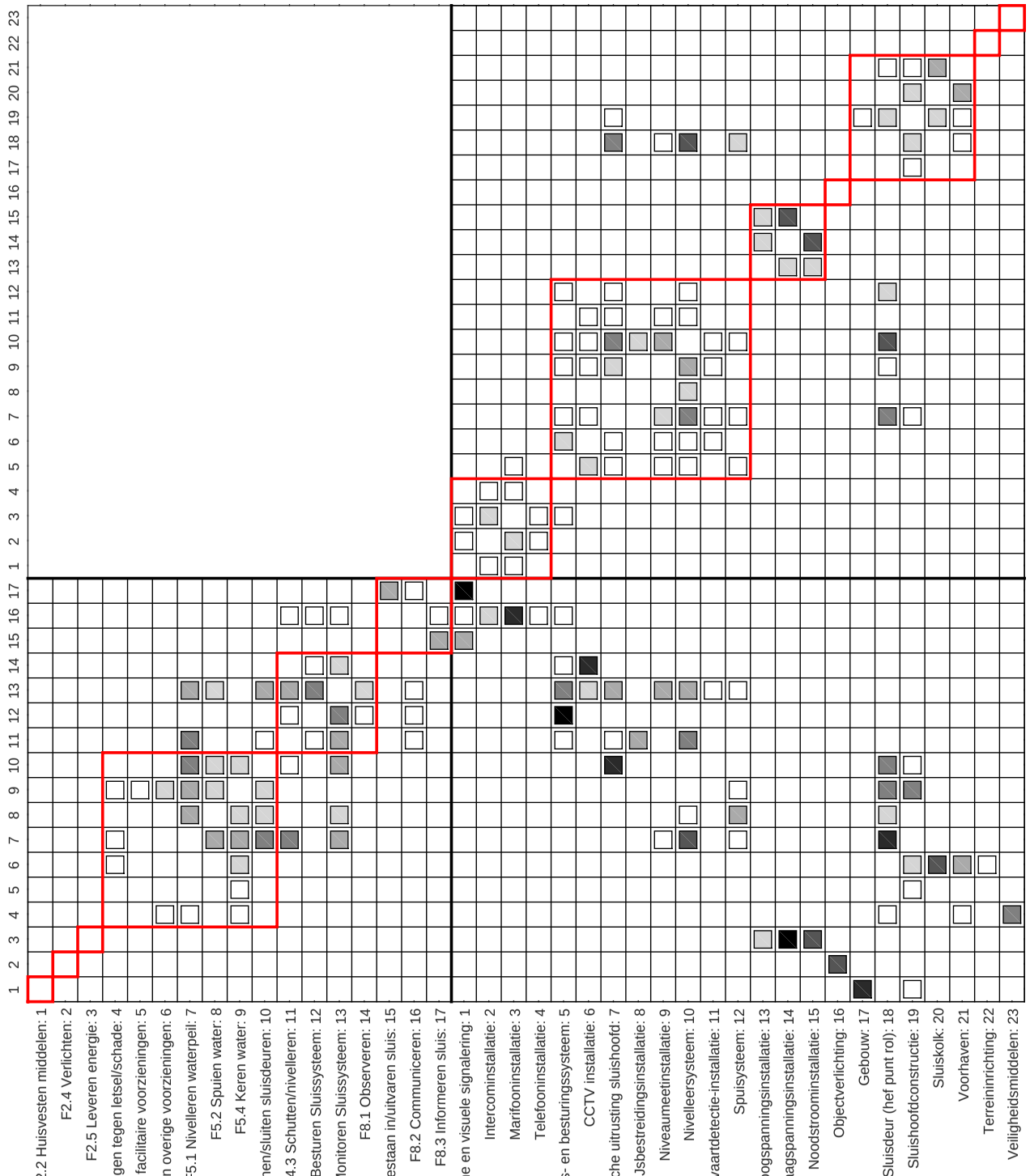


Figure 5.3: The Σ multi-domain matrix (ΣMDM) based on FMECA files.

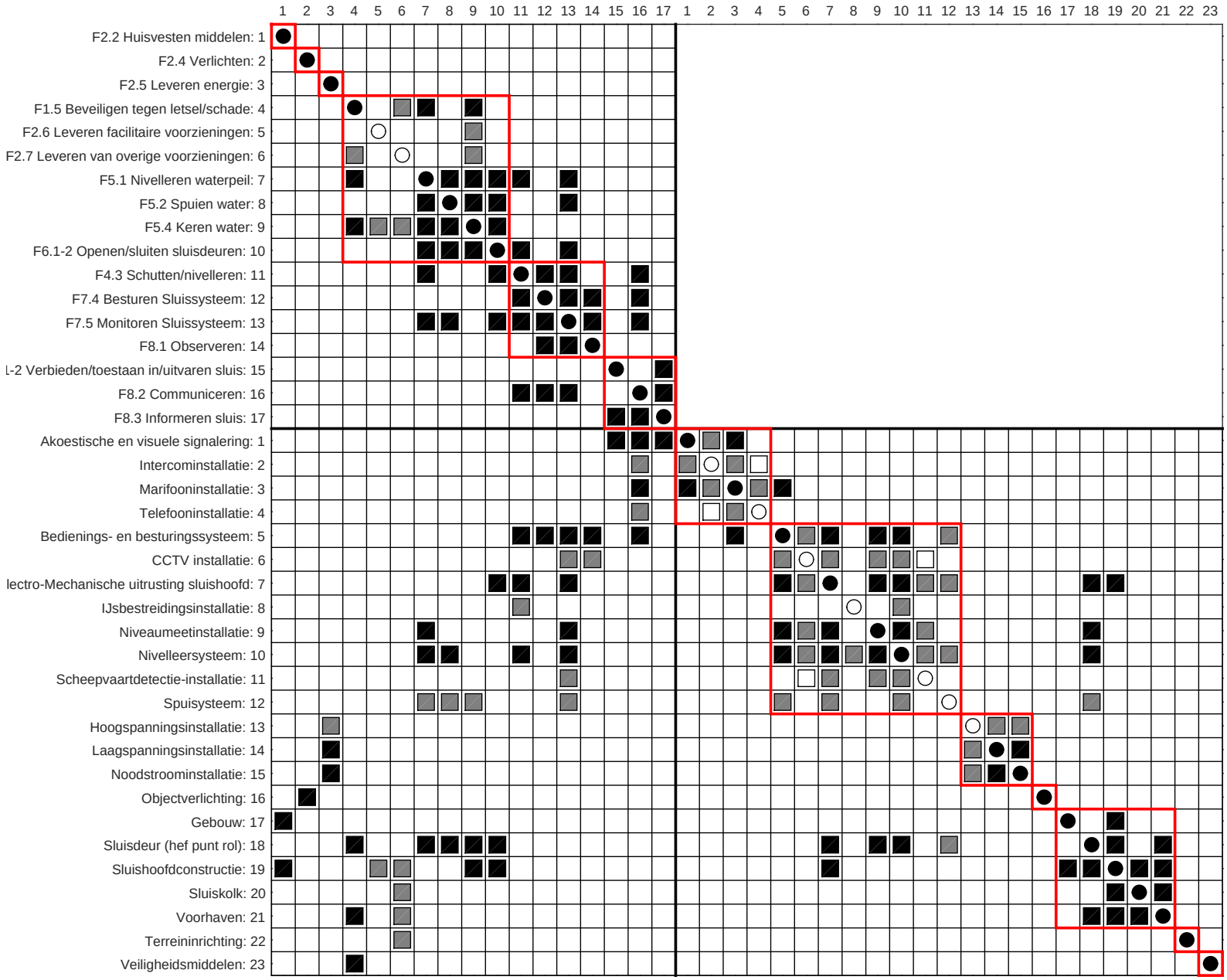


Figure 5.4: The variety multi-domain matrix (VMDM) based on FME(C)A data.

Chapter 6

Lock family reliability and availability

Prime candidates for standardization are components which are key to the function of the lock and have a significant impact on RA, LCC and RTC. Through standardization one can pursue a highly predictable RA and a short mean time to repair (MTTR). Chapter 5 argues that a low innovation rate and a low dependency on situation specific factors are preconditions for standardization as well. This chapter presents an algorithm for the detection of components which are key to the functioning of the system.

6.1 Motivation

In this study, failure, reliability, and availability of a system are defined as follows (Birolini [2007]):

- **Failure** occurs when the system stops performing its required function at a point in time.
- **Reliability** of a system is the probability that no operational interruptions will occur during a stated time interval. Note that redundant parts within the system may fail without effecting the reliability of the system as a whole, when such parts can be repaired without operational interruptions. So redundancy of components is advantageous for the system reliability.
- **Availability** is the probability that the system can perform its required function under given conditions at a stated time interval.

Note the recurring words 'system' and 'time' within these definitions. Failure, reliability and availability are characteristics which are a result of the interplay between the physical (P) and operational (O) domains. As such, the RA of lock components can be evaluated using historical data about lock component failures and repair times.

However, using historical data in this study is not an option. First and foremost, such data is not (easily) accessible at RWS. Secondly, the 37 existing locks were mostly build during the 1930's. Technology has seen a tremendous advancement since. Therefore, it is questionable if historical failure and repair data of those locks can be used to determine components which have a high impact on RA of modern locks. For example, corrosion of concrete rebar used to be a significant problem, however nowadays this is no longer an issue.

	c1	c2	c3	c4	c5	c6	$w_{\mathrm{f},i}$
f1		9			3		9
f2	3		9				3
f3		3		9	1		3
f4				9			1
f5	9		3			1	9
$w_{\mathrm{c},j}$	90	90	54	36	30	9	

Table 6.1: Example quality function deployment table.

Third of all, a lock is a multi-functional system, i.e. it has three main functions, namely the 'Passeerfunctie', the 'Keerfunctie' and the 'Peilbeheerfunctie', and many sub-functions. Therefore, it is not trivial to determine when a lock stops performing its required function, i.e. when a lock is in the state 'failure'. For example, a lock could be in a state in which it can perform two out of the three main functions. Implying that the lock is unavailable with respect to only one function. As a result, the RA of a lock will differ with respect to the three main functions and many sub-functions. Birolini [2007] states that a numerical statement on RA must always be accompanied by the definition of the required function, operating conditions, and time interval.

The same holds for the locks components. In Chapter 4 it has been shown that lock components contribute to the fulfillment of multiple functions. The RA of a component may differ with respect to the different functions it contributes to. Moreover, the RA of a single component with respect to a single function is not representative for the RA of the complete lock. Therefore, it is suggested to use the function-component MDM to determine which components cause most loss of functionality in case of failure. To this end, a specific algorithm has been developed.

6.2 Quality function deployment

The developed algorithm determines which components cause most loss of functionality in case of failure based on the function-component MDM structure. However, not all sub-functions are equally important for the lock. Moreover, a function is usually fulfilled by multiple components and not all components are equally important in fulfilling that particular function. Therefore, each function and component is assigned a weight of importance w_{I}, which is used to guide the search algorithm towards more important functions/components. These weights are determined using a method referred to as quality function deployment (QFD). In this section, QFD is only briefly explained. For more, information on QFD the reader is referred to Bahill et al. [1993], Govers [1996] and Ficalora and Cohen [2009].

QFD was originally developed to introduce the idea of quality in the conceptual design phase. In this study, QFD is used to introduce the idea of RA in the functional and conceptual design domains. In Table 6.1 an example QFD is shown. The rows are labeled with the system functions, the columns are labeled with the system components. Note that the shown QFD table is the transposed of the DM shown in Figure 4.1.

In Table 6.1, each function-component dependency is assigned a weight of either 1, 3 or 9 by experts. The higher the weight, the more important component j is for the fulfillment of function i. Similarly, each function i is assigned a weight of importance $w_{\mathrm{f},i}$ of either 1, 3 or 9. The higher the weight the more important function i is for the functioning of the complete system. Subsequently, the component weight of importance $w_{\mathrm{c},j}$ can be calculated using Equation (6.1).

$$w_{\mathrm{c},j} = \sum_{i=1}^{N} QFD(i,j) \cdot w_{\mathrm{f},i} \tag{6.1}$$

herein, $w_{c,j}$ is the weight of importance of component j with respect to the functioning of the complete system, N is the number of system functions, $QFD(i,j)$ is the dependency weight of function i and component j, and $w_{f,i}$ is the weight of importance of function i.

In the example presented in Table 6.1 components c1 and c2 are the most important components with respect to the functioning of the complete system. As such, the RA of c1 and c2 is expected to have the highest influence on the RA of the complete system. However, c1 and c2 do not fulfill the system functions by themselves, i.e. they are functionally dependent on other components. Therefore, components which ensure that c1 and c2 function properly, could have a significant influence on the system RA as well. Those components are not highlighted by the QFD analysis. As such, it is also required to study the system dependency structure to find all components which could have a significant impact on the RA of the complete system.

6.3 RA ranking

Google uses a ranking algorithm (Page et al. [1999]) which determines the importance of websites with respect to a certain keyword based on the hyperlink structure of the world wide web. The algorithm can be personalized based on the surfers interests and browsing history.

In this study the ranking algorithm is used to determine the importance of components/functions with respect to RA based on the dependency structure between components/functions denoted by R scores. The QFD scores will be used to 'personalize' the algorithm, ensuring that the algorithm finds components which are important with respect to RA.

Table 6.2 shows the QFD and the R scores for both the functions and the components. The tables are normalized such that total sum of rank in each column equals 1. Note that the functions f2 and f3 significantly increase in relative importance if one uses the ranking algorithm, similar holds for components c5 and c6.

The reason for this increase in importance is seen in Figure 6.1, which shows a modified version of the function-component multi-domain matrix (MDM) previously seen in Chapter 4. The color of the circles along the diagonal of the MDM reflect the R scores, the darker the color, the more important the function/component. The color of a dependency mark at position i,j is based on the product of the R scores of elements i and j ($R_i \cdot R_j$). The higher the product, the more important the dependency, the darker the color.

Table 6.2a shows that functions f1 and f5 obtained the highest QFD scores. Looking at Figure 6.1 it is seen that functions f1 and f3 are dependent on one another. As such, the ranking algorithm will make function f3 more important since function f1 depends on it. Similarly, function f5 dependents on function f2, therefore f2 obtains a high R score.

Table 6.2b shows that components c1 and c2 have the highest QFD scores. Figure 6.1 shows that components c1 and c6 are dependent on one another, therefore component c6 increases in importance. Equally, component c2 depends on component c5, therefore component c5 increases in importance.

Furthermore, Figure 6.1 shows that components c1 and c2 are most important with respect to the functionality of the system. If one of these components fails, or if the interface between these components fails, the system looses the functions f2 and f5, which both have a high importance score.

6.4 Data processing

In Chapter 5 FME(C)A files are used to generate the lock family function-component MDM. Within these files components are linked to functions. Moreover, the importance of a component

	QFD score	R score
f1	0.36	0.23
f2	0.12	0.20
f3	0.12	0.22
f4	0.04	0.08
f5	0.36	0.27
total	1.00	1.00

(a)

	QFD score	Ranking score
c1	0.29	0.21
c2	0.29	0.21
c3	0.17	0.17
c4	0.12	0.15
c5	0.10	0.14
c6	0.03	0.12
total	1.00	1.00

(b)

Table 6.2: Normalized QFD and R scores of the functions (a) and the components (b).

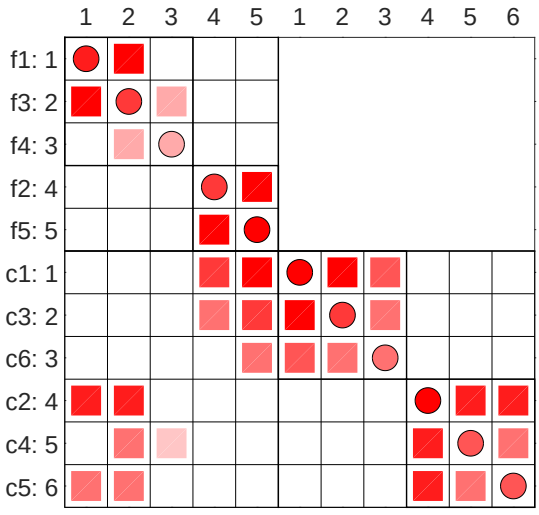


Figure 6.1: Example RA risk multi-domain matrix.

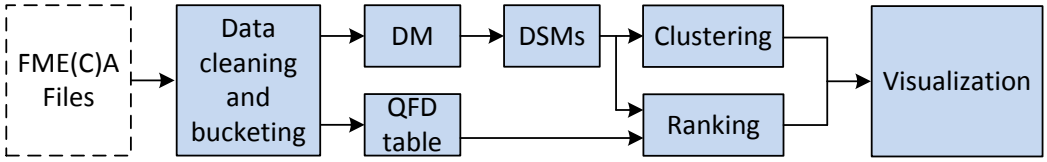


Figure 6.2: The seven steps of the developed RA-analysis.

with respect to a function is rated as: 'has no influence', 'has little influence', or 'has influence' on the performance of the function to which the component contributes. These strings are converted to the QFD dependency weights: 0, 3 or 9 respectively. Additionally, the files contain information on whether a function is required for the fulfillment of the three main functions: 'Passeerfunctie', 'Keerfunctie', and 'Peilbeheerfunctie'. A function which contributes to all three main functions is assigned a weight of 3, a function which contributes to two main functions is assigned a weight of 2, and a function which contributes to one main function is assigned a weight of 1. Functions which do not contribute to any of the main functions are not considered.

Figure 6.2 shows the seven steps of the developed RA-analysis. The first step consists of filtering, cleaning and bucketing the data such that the components and functions are formulated on a similar level of abstraction. Moreover, many typos and inconsistencies in function and component names are corrected. Often multiple variants of the same function/component name are observed, for example 'afvoer regenwater' and 'afvoeren regenwater'.

The filtered, cleaned and bucketed data is used to generate a design matrix (DM) and a quality function deployment (QFD) table for each lock separately. The DM is used to create a function and a component DSM following the method described in Chapter 4. The MDMs are added up into a ΣDSM. The QFD tables are merged into a ΣQFD, in which the scores are averaged over the number of FME(C)A files. The resulting data is analyzed using a Markov clustering algorithm (Van Dongen [2008]) and the ranking algorithm. The final step consist of visualizing the results as shown in Figure 6.1.

6.5 Results

Table 6.3 shows the QFD and R scores based on the FME(C)A files of seven locks. The five functions/components with the highest rank are shaded. Looking at Table 6.3a it is seen that the QFD analysis and the ranking algorithm yield a different top five. Note that the function 'F7.5 Monitoren sluissysteem' is indicated as an important function by the ranking algorithm and not by the QFD analysis. In Figure 6.3 it is seen that function F7.5 has many dependencies with other important functions. As such, the function itself also becomes more important. Similar holds for the function 'F5.4 Keren water'.

The FME(C)A files contain many sub-functions which are placed within the 'F2.7 Leveren van overige voorzieningen' bucket. The combined weight of these sub-functions yields function F2.7 a place in the top five QFD ranking, despite the fact that the sub-functions are relatively unimportant with respect to the three main functions. However, Figure 6.3 shows that function F2.7 has no dependencies with other QFD-top-five functions. As such, it is rated as less important by the ranking algorithm.

Similarly, the function 'F2.5 Leveren energie' is not in the ranking-top-five since it has no dependencies with other functions. However, a source of energy is required to fulfill two of the three main

functions. Figure 6.3, shows that the components 'Laagspanningsinstallatie', 'Hoogspanningsinstallatie, and 'Noodstroomvoorziening' contribute to the fulfillment of function F2.5. Those three components do not contribute to any other function. Though, common sense would suggest that one would require an energy source to open and to close the doors, i.e to fulfill function F6.1-2. However, within the FME(C)A files, components are only related to functions to which they directly contribute, the files contain no information on indirect relations. As a result, no indirect function-component dependencies are shown within these figures. As such, the MDMs presented in this study do not show the full lock family structure, which is important to keep in mind while interpreting the results.

Table 6.3b shows that component 'Elec.- Mech. uitrusting sluishoofd' is ranked as most important component by both the QFD as the RA analysis. The 'Nivelleersysteem' has become more important than the 'Spuisysteem' this is a result of the fact that the 'Nivelleersysteem' contributes to the two main functions 'Passeerfunctie' and 'Peilbeheerfunctie'. The 'Spuisysteem' only contributes to the main function 'Peilbeheerfunctie'.

It is observed that the QFD score of component 'Elec.- Mech. uitrusting sluishoofd' is five times higher than that of the second most important component 'Spuisysteem'. Electrical and control components are denoted relatively unimportant. Therefore, it is thought that the FME(C)A files are made from a mechanical engineering perspective. As such, those components received much attention within the FME(C)A analysis. As a result, the mechanical components obtained high QFD scores. Moreover, Figure 6.3 shows that the mechanical components have on average more dependencies than the electrical and control components which is a result of the fact that only direct function-component dependencies are present within the DM. As a result of the high QFD scores and high number of dependencies, the RA algorithm also ranks the mechanical components as most important.

Despite the mechanical engineering perspective of the FME(C)A files and the incomplete lock family structure, it is possible to draw some preliminary conclusions. The component 'Elec.- Mech. uitrusting sluishoofd' is ranked most important with respect to the functionality of the system. Moreover, it has important dependencies with the components 'Nivelleersysteem' and the 'Sluisdeur (hef, punt, rol)'. Looking back at the variety MDM shown in Figure 5.4, it can be seen that all three components are always present within a lock. Therefore, the 'Elec.- Mech. uitrusting sluishoofd' is probably a suitable candidate for standardization. Since it has a high impact on lock functionality in case of failure, and is always present within a lock.

6.6 Conclusion

A lock is a multi-functional system. The RA of a lock component may differ with respect to the different lock functions. Therefore, an algorithm is designed which determines which components cause the most loss of functionality in case of failure. Those components are likely to have a significant impact on the RA of the complete lock. A benefit of this approach is that it analyses the functional dependency structure of the lock. As such, RWS can detect RA risks early in the design process without any knowledge on the physical form of the lock.

The first results indicate that functions 'F6.1-2 Openen/ sluiten sluisdeuren' and 'F7.5 Monitoren sluis systeem' and components 'Bewegingswerken' and 'Sluisdeur' may have a significant impact on RA, and may be suitable candidates for standardization. However, the used data does not cover all the functions and components which could be present within a lock. Moreover, the data is probably made from a mechanical engineering perspective which resulted in relatively high QFD scores for mechanical components.

As such, to obtain a conclusive result, it is required to model the complete lock family structure and create a QFD containing all lock functions and components which could be present in a lock.

	QFD score	R score
F1.5 Beveiligen tegen letsel/schade	0.03	0.03
F2.2 Huisvesten middelen	0.03	0.03
F2.4 Verlichten	0.02	0.01
F2.5 Leveren energie	0.07	0.04
F2.6 Leveren facilitaire voorzieningen	0.02	0.02
F2.7 Leveren van overige voorzieningen	0.09	0.06
F4.1-2 Verbieden/toestaan in/uitvaren sluis	0.01	0.02
F4.3 Schutten/nivelleren	0.07	0.07
F5.1 Nivelleren waterpeil	0.09	0.12
F5.2 Spuien water	0.09	0.09
F5.4 Keren water	0.03	0.09
F6.1-2 Openen/sluiten sluisdeuren	0.24	0.16
F7.4 Besturen Sluissysteem	0.04	0.05
F7.5 Monitoren Sluissysteem	0.05	0.11
F8.1 Observeren	0.04	0.03
F8.2 Communiceren	0.03	0.03
F8.3 Informeren sluis	0.03	0.03
total	1.00	1.00

(a)

	QFD score	R score
Akoestische en visuele signalering	0.01	0.01
Bedienings- en besturingssysteem	0.03	0.05
CCTV installatie	0.02	0.03
Elect.-Mech. uitrusting sluishoofd	0.45	0.26
Gebouw	0.01	0.01
Hoogspanningsinstallatie	0.03	0.02
IJsbestreidingsinstallatie	0.00	0.01
Intercominstallatie	0.00	0.01
Laagspanningsinstallatie	0.04	0.03
Marifooninstallatie	0.00	0.01
Niveaumeetinstallatie	0.01	0.04
Nivelleersysteem	0.06	0.10
Noodstroominstallatie	0.01	0.03
Objectverlichting	0.00	0.01
Scheepvaartdetectie-installatie	0.02	0.02
Sluisdeur (hef punt rol)	0.08	0.11
Sluishoofdconstructie	0.02	0.05
Sluiskolk	0.03	0.03
Spuisysteem	0.09	0.06
Telefooninstallatie	0.00	0.01
Terreininrichting	0.01	0.01
Veiligheidsmiddelen	0.01	0.01
Voorhaven	0.03	0.03
total	1.00	1.00

(b)

Table 6.3: Normalized QFD and R scores of the functions (a) and the components (b). The top five rated functions/components are shaded.

Figure 6.3: The family RA-risk multi-domain matrix based on FME(C)A data.

Chapter 7

Case study: Lock Eefde reliability and availability

In this study the focus has primarily been on the functional (F) and conceptual design (C) domains of the life cycle multi-domain matrix, depicted in Figure 3.1. The function-component multi-domain matrix (MDM) has been used as an abstract representation of the design of a lock. One may argue to what extend the function-component MDM represents the physical dependency structure of a lock, i.e. the dependencies between components within the physical (P) domain, and to what extend the results obtained from analyzing the function-component MDM with respect to modularization, RA and LCC, are representative for a real lock.

To this end, Dijkstra [2015] conducted a case study for lock Eefde in which Dijkstra compared a component DSM describing functional dependencies with a component DSM describing physical dependencies. The component DSM describing the functional dependencies is created following the method described in Chapter 4 using GRIP and FME(C)A files as input data. The component DSM describing the physical dependencies is created by reviewing design documents and interviews with lock experts.

He analyzed the physical dependency structure to identify components and component dependencies with a significant impact on reliability and availability following a method inspired by the work of Brady [2002]. Brady developed a method for the identification of technology risk in the NASA pathfinder robot. This chapter provides a summary of the work of Dijkstra [2015].

7.1 Functional vs. physical dependencies

The life cycle structure matrix (Figure 3.1) illustrates that different domains play a dominant role at different stages of the lock life cycle. In each stage the lock dependency structure can be described at a different level of abstraction.

In the functional (F) and conceptual (C) design domains it suffices to only look at the functional dependency structure. Though, during embodiment design (E) and detailed design (D) it is required to look at the physical dependencies between components, i.e. the physical form of the dependencies between components. One needs this information to integrate the different components and subsystems into a single system.

In this section the relationship between the functional and physical dependency structure is illustrated using an example. Assume that a conceptual design is made in which three components c1, c2 and c3 have to fulfill the function f1: 'Connect point A with point B'. The resulting functional dependency structure of this conceptual design is shown in Table 7.1a, which shows the

	f1	c1	c2	c3
f1	-			
c1	x	-	x	x
c2	x	x	-	x
c3	x	x	x	-

(a)

	c1	c2	c3
c1	-	x	
c2	x	-	x
c3		x	-

(b)

	c1	c2	c3
c1	-	x	x
c2	x	-	
c3	x		-

(c)

Table 7.1: (a) The function-component MDM, (b) the structural DSM of configuration 1, (c) and the structural DSM of configuration 2.

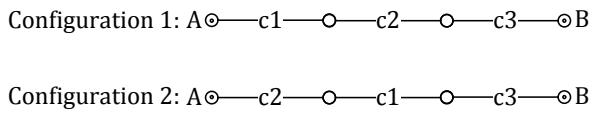


Figure 7.1: Two possible structural configurations to physically connect point A with point B using three components.

function-component multi-domain matrix. The component DSM, in Table 7.1a, is generated via the method described in Chapter 4 and shows that all three components are mutually functionally dependent.

After a conceptual design is created, engineers can start with embodiment design. In this step engineers have to determine the actual configuration of components c1, c2 and c3 that fulfill function f1. For example, Figure 7.1 shows two possible structural configurations to physically connect point A with point B. In configuration 1, component c1 is physically attached to component c2, i.e. there is a structural dependency between components c1 and c2. In turn, component c2 is connected to component c3, i.e. there is a structural dependency between components c2 and c3. These dependencies are shown in DSM shown in Table 7.1b.

In configuration 2, component c2 is physically connected with c1, i.e. their is a structural dependency between components c1 and c2. Component c1 is also physically connected to component c3, i.e. their is a structural dependency between components c1 and c3. These dependencies are shown in the DSM shown in Table 7.1c.

The two physical configurations yield different structural DSMs, i.e. a different physical configuration yields a different system structure. However, marks shown in the structural DSMs are subsets of the marks shown in the functional component DSM (Table 7.1a). As such, the functional dependency DSM structure defines the design space in which the physical dependency structure should lie. In other words, a conceptual design should define a design space in which a feasible physical design can be found which meets all stated requirements.

A public tender contains requirements and functions which are related to objects (components). As such, the tender describes a design space in which the contractors should be able to find a feasible physical design which meets all stated requirements. However, RWS has no tooling to analyze and visualize the design space which is described by a tender. By analyzing and visualizing the design space using DSM techniques, RWS can verify whether the tender describes the desired design space and if it is possible for the contractor to find a feasible design.

7.2 The physical DSM

In a period of several months Dijkstra [2015] has reviewed design documentation of lock Eefde in search of a suitable component decomposition and physical dependencies between the components. He focused his efforts on finding the following four types of dependencies:

- **Energy**: dependencies indicate energy exchange between a pair of components.
- **Information**: dependencies indicate information exchange between a pair of components.
- **Spatial**: dependencies indicate that a pair of components are physically adjacent, i.e. altering the dimensions of one component will influence the dimensions of the other component.
- **Location**: dependencies indicate that a pair of components has to be in proximity with respect to one another.

Subsequently, Dijkstra improved upon his results by use of interviews with system experts. The results of his efforts are shown in Figure 7.2, where yellow, blue, red and purple marks indicate energy, information, spatial and location dependencies respectively.

Figure 7.2 shows seven component clusters, of which the first is a so called 'bus' cluster. A bus is a common phenomenon in DSM analysis (Browning [2001]). The term bus originates from the bus component of a computer which is a central communication system that transfers data between the various components inside a computer.

Bus components typically serve as coordination hubs within a system and therefore have many dependencies with many different components. In Figure 7.2 it can be seen that the 'Besturingssysteem', the 'Laagspanningsinstallatie' and the 'Noodstroomvoorziening' are part of the bus and have many energy and information dependencies with other components. The more dependencies a component has, the more complex its design and manufacturing process becomes. As such, bus components require special attention and sufficient resources during the specification and manufacturing phases of the lock life cycle.

The complexity of designing and manufacturing bus components can be reduced by applying (interface) standardization. Moreover, it would simplify the process of integrating many locks components, reduce the chance of design flaws, and leverage development costs over multiple locks.

Looking at the component clusters, it is observed that the clusters contain mechanical components, civil components, components related to the pre-port ('Voorhaven'), operating components, communication components, and pre-pock ('Voorsluis') components. The contents of these clusters is reasonably consistent with the clusters found in the family function-component multi-domain matrix presented in Chapter 5. Therefore, designing the lock development platform based upon the analysis of the function-component dependency structure found in locks seems a valid approach.

Interesting is the relatively small number of spatial dependencies, arguing that from a mechanical/civil perspective a navigation lock is a simple system. On the other hand, from an energy/information management perspective a lock is relatively complex. If in the future locks are further automated the information/energy dependencies will become even more complex.

7.3 Reliability and availability

In the near future lock Eefde is to be expanded with a second 'kolk'. This second 'kolk' has to be integrated with the already existing system. Dijkstra [2015] analyzed the lock dependency structure of the current lock to identify components with a significant impact on RA. These components should receive special attention during the design, construction and integration of the second 'kolk' to ensure the new system will have a high RA.

Dijkstra [2015] followed a method inspired by the work of Brady [2002]. Brady developed a method to determine technology risk in the design of the NASA pathfinder robot, i.e. to determine the risk of multiple integrated technologies not working properly. Dijkstra modified this method to identify components with a significant impact on RA.

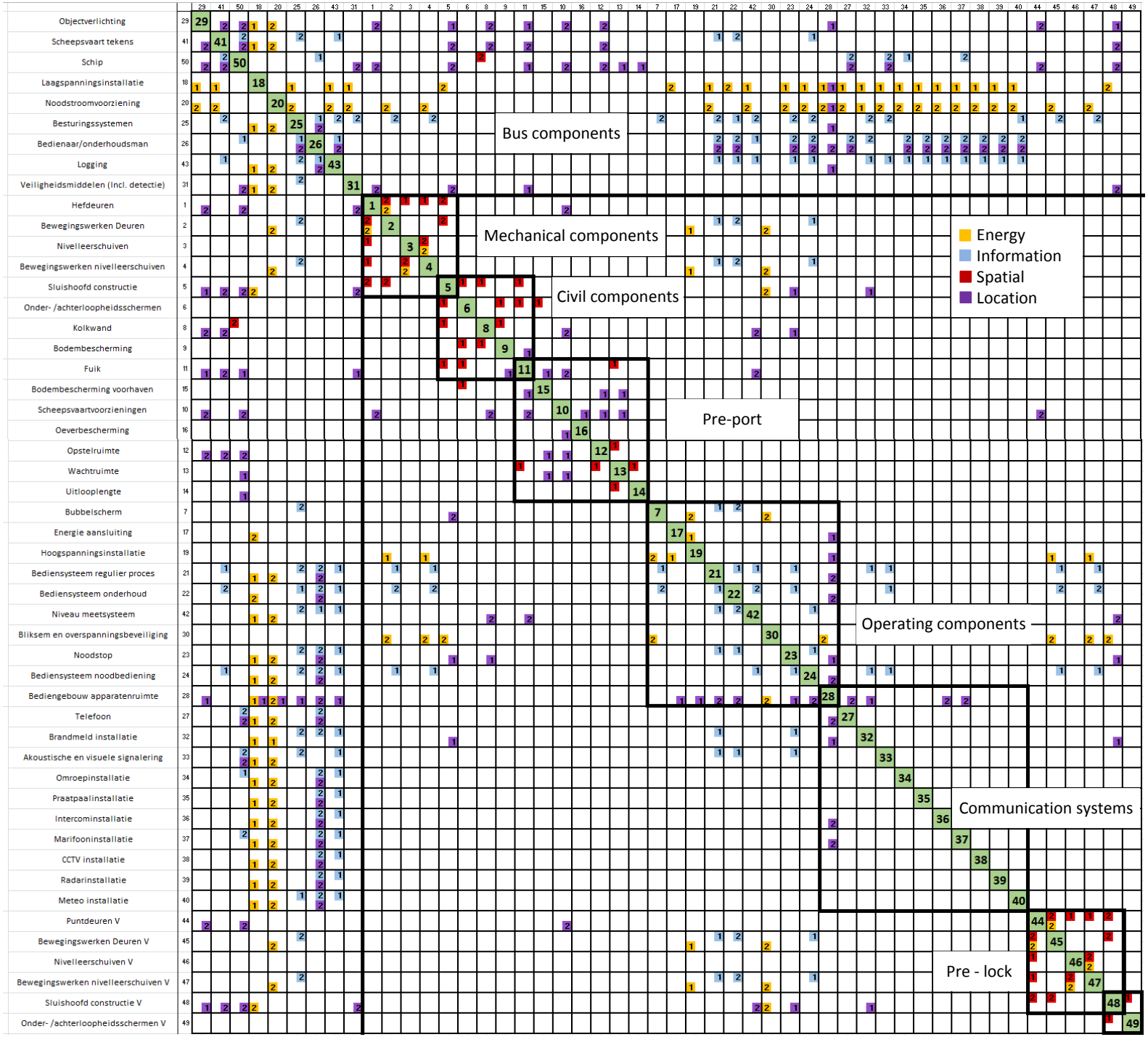


Figure 7.2: Physical DSM of lock Eefde

The RA-analysis method of Dijkstra [2015] consists of the following four steps:

1. Determine an R- and an A-impact factor for all individual components, i.e. determine the impact on R and A if a certain components fails. The R-impact factor is a function of the component failure frequency. The A-impact factor is a function of the component failure frequency and the component mean time to repair. Dijkstra based these factors on fault tree analysis files found in RINK, created by Arcadis. These files were mainly focused on structural failure of components and are based on the *current condition* of lock Eefde. As such, failure frequencies and MTTR values may differ from a complete new lock. For those components which were not present within the data, e.g. the 'Operator'and the 'Schip', Dijkstra estimated the R- and A- impact factors himself.
2. Aggregate the four dependency types shown in Figure 7.2 into a single dependency denoting the interaction strength between components.
3. Calculate the R- and A-impact dependency matrix. An entry at position i, j is the result of multiplying the interaction strength between components i and j, with the R(A)-impact of component i and the R(A)-impact factor of component j.

Regarding step 3, Dijkstra [2015] did not make use of the RA ranking algorithm presented in Chapter 6 but followed the method of Brady [2002].

The results of Dijkstra's effort with respect to reliability are presented in Figure 7.3. The R-impact matrix shows that the 'Besturingssystemen' has high impact with the 'Logging', the 'Bewegingswerken Deuren', and the 'Beweginswerken nivelleerschuiven'. The 'operator' has a high impact dependency with the 'CCTV installation'.

Within the mechanical component cluster two high impact dependencies are observed between the 'Hefdeuren' and the 'Bewegingswerken Deuren' and between the 'Bewegingswerken nivelleerschuiven' and the 'Nivelleerschuiven'. The R-impact factor has not been compensated for redundancy. As such, the impact of the 'Nivelleerschuiven' is probably less critical than depicted in Figure 7.3.

The results of Dijkstra's efforts with respect to availability are presented in Figure 7.4. Note the the critical dependencies are shifted with respect to the reliability results shown in Figure 7.3. The A-impact matrix shows that the mechanical and civil component clusters contain multiple high impact dependencies, which could be due to the high mean time to repair times of these components. In particular, the dependency between the 'Beweginswerken deuren' and the 'Hefdeuren' have a high A-impact score. This result is consisted with the results obtained via the RA family analysis discussed in Chapter 6. Therefore, the 'bewegingswerken' are considered to be a prime candidate for standardization.

Interestingly, the 'Besturingssysteem' and 'Laagspanningsinstallatie' have many dependencies with a relatively low A-impact. This could be due to the fact that the fault tree analysis files primarily focused on structural failure of components, not taking into account the mechanisms via which control and energy systems can fail.

Contrary, the 'operator' has multiple dependencies with a relatively high impact on A. Indicating the need for proper training of the operators and standardization of operating procedures. However, it most be noted that Dijkstra estimated the A-impact factor for the 'operator' himself.

7.4 Comparison

The physical dependency structure of lock Eefde should be a subset of the functional dependency structure as explained Section 7.1. Therefore, Dijkstra [2015] also created a function-component

design matrix based upon GRIP and FMECA data. Subsequently, he used the method described in Chapter 4 to generate a component DSM denoting the functional dependencies between components.

Dijkstra [2015] created the comparison DSM, in which he combined the functional and physical dependency structure, shown in Figure 7.5. A one (light blue) indicates a dependency which is only found in the functional structure, a two (blue) indicates a dependency which is only found in the physical structure and a three (deep blue) indicates a dependency which is found in both the functional and physical structure.

Looking at the component clusters, it can be seen that there are many dependencies which are only found in the functional DSM. This is particularly evident in the 'Communication components' cluster. This due to a leveling issue, i.e. almost all communication components are linked to the general function 'F8.2 Communiceren'. Despite the fact that a 'Telefoon' has a different function than a 'Intercominstallatie'.

By linking both the 'Telefoon' and the 'Intercominstallatie' to the function 'F8.2 Communiceren', RWS defines a functional relation between these two components which is part of the conceptual design space. Allowing the contractor to create a physical dependency between the 'Telefoon' and the 'Intercominstallatie'. By linking the telefoon to function 'F8.2.3 Bieden telefonie' and the 'Intercominstallatie' to function 'F8.2.2 Bieden audiocommunicatie op de sluis', RWS can remove the functional dependency between these two components. As a result, the contractor is not be allowed to create a physical dependency between the 'Telefoon' and the 'Intercominstallatie'.

As such, by carefully managing the abstraction levels on which the functions and components are formulated and are linked to one another, RWS can inflate or deflate the design space, i.e. RWS can increase or decrease the design freedom a contractor has.

In the bus, the 'Laagspanningsinstallatie' and the 'Noodstroomvoorziening' have many dependencies which are only found in the physical DSM. This is due to the fact that the GRIP data only contains direct function component relations, as such indirect functional dependencies are not found. For example, in GRIP no functional relation is found between the 'Laagspanningsinstallatie' and the 'Besturingssysteem'. Despite the fact that the 'Besturingssysteem' requires energy to operate.

Contrary, the 'Besturingssysteem' has multiple unexpected dependencies that are only found in the functional DSM. For example, a functional dependency between the 'Besturingssysteem' and the 'Wachtruimte'. This is a result of the fact that these components were all linked to the top function 'Passeren scheepvaart'.

By looking at Figure 7.5 globally, it can be observed that most dependencies that are found both in the functional and in the physical dependency structure, fall within a components cluster or in the bus. This indicates that both dependency structures share a similar underlying structure.

Therefore, it is postulated that if the functions and components of the design matrix are formulated on a similar level of abstraction, and one includes indirect functional relations, the generated functional component dependency structure will neatly define the design space in which the physical dependency structure lies.

7.5 Conclusion

The DSM methods applied for the Eefde case and the lock family analysis using FME(C)A files provided insightful results regarding the component dependencies. The DSM shows that most of the complexity of a lock Eefde is due to energy and information dependencies. By clustering the DSM distinct component clusters are found of which the 'bus' cluster is particular interesting. Bus

components (e.g. 'Besturingssysteem' and 'Laagspanningsinstallatie') could benefit from interface standardization.

The RA analysis approach of Dijkstra indicated that the mechanical and civil component clusters show multiple high impact dependencies on both reliability as availability, which could be due to the high failure frequencies of the mechanical components and the high mean time to repair times of the civil components. In particular, dependencies between 'Beweginswerken' and 'Hefdeuren' and are denoted as high impact dependencies with regard to reliability and availability. In the bus, 'Besturingssysteem' has multiple high impact dependencies with respect to reliability, while 'Schip' and 'Kolkwand' have a high impact dependency on availability. The 'Bedienaar' has many dependencies with a relatively high impact on both reliability and availability. However, it should be noted that Dijkstra estimated the R(A)-impact factor for the 'operator' himself.

Based upon the comparison of the component DSM showing the functional dependencies of lock Eefde with the component DSM showing the physical dependencies, it is postulated that by creating a design matrix in which the functions and components are formulated on a similar, sufficiently detailed level of abstraction, RWS can define a design space in which the physical dependency structure lies. As such, by tuning the level of abstraction on which the functions and components are formulated RWS can increase of decrease the design freedom a contractor has.

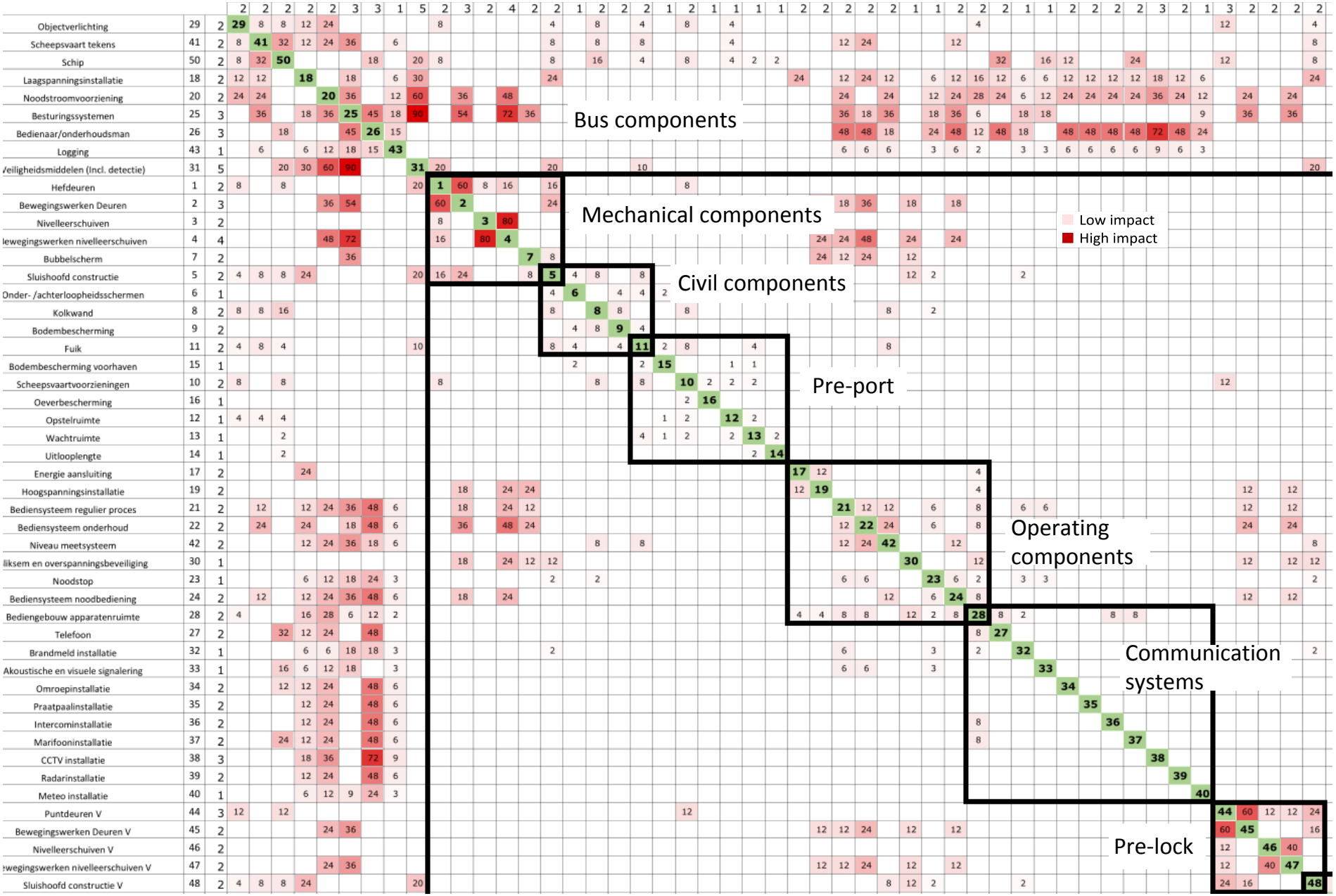


Figure 7.3: The reliability impact matrix of lock Eefde based on failure frequencies of components.

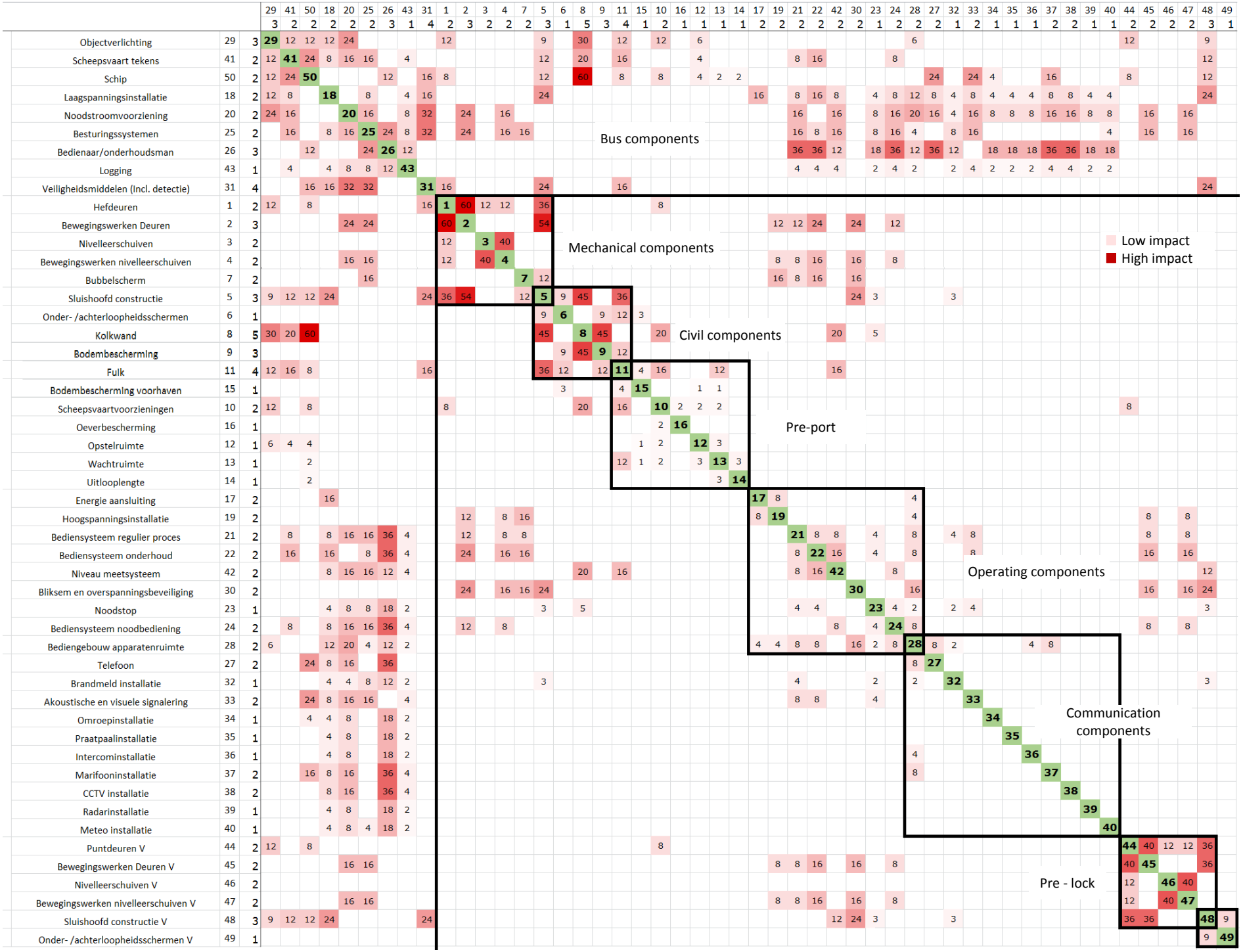


Figure 7.4: The availability impact matrix of lock Eefde based on failure frequencies and mean time to repair of components.

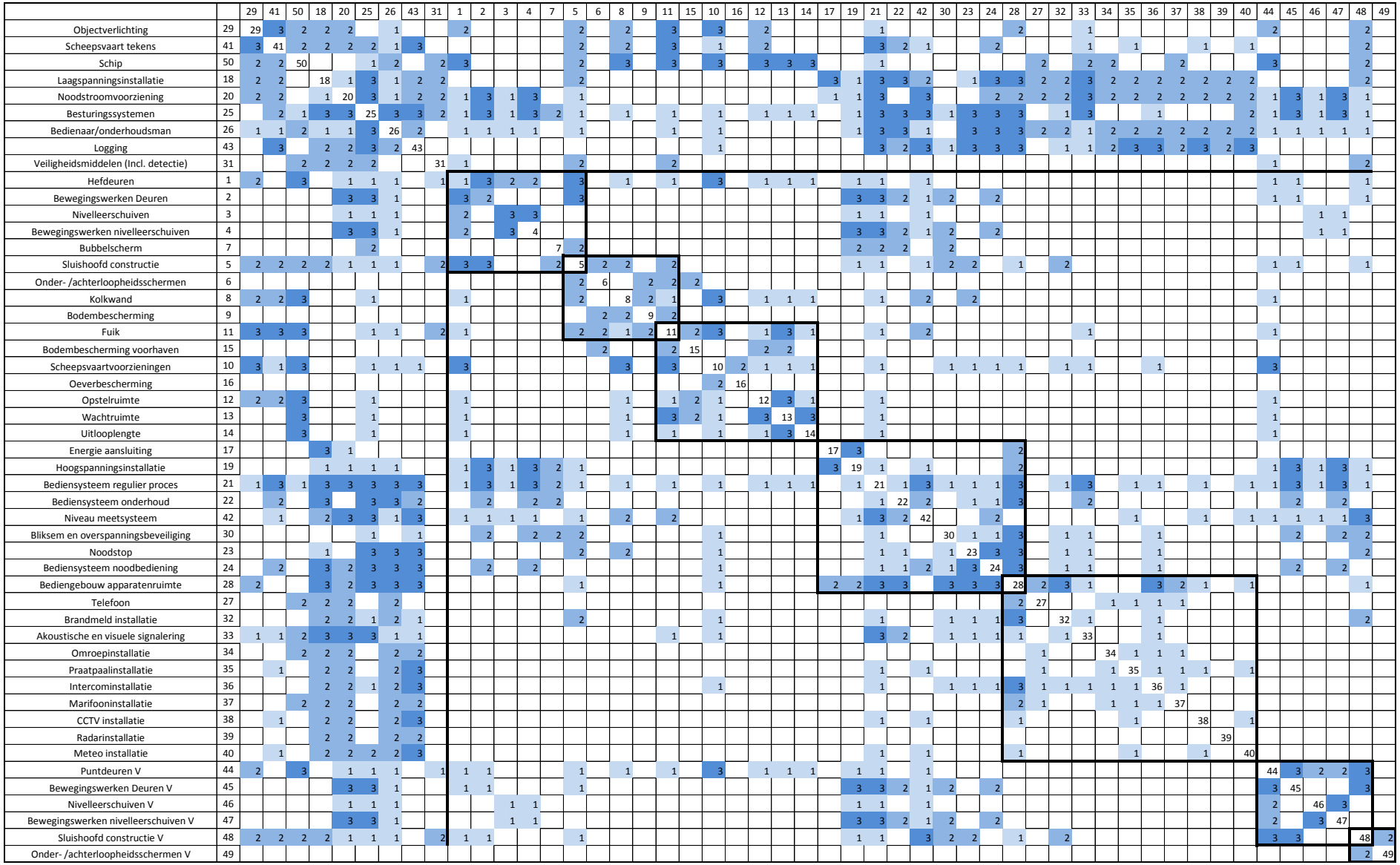


Figure 7.5: Comparison of functional and physical dependence structure of lock Eefde

Chapter 8

Lock construction

One of the goals of MWW is to reduce the uncertainty in time and costs of construction projects. A construction project resides in the manufacturing domain (M) depicted in Figure 3.1, and can be decomposed into a number of activities which are performed in parallel and/or sequentially. In this study we are particular interested in the risk of exceeding construction time and cost (RTC). Risk is not the same as uncertainty, while uncertainty is a potential, unpredictable, unmeasurable and uncontrollable outcome, risk is a consequence of action taken in spite of uncertainty (Antunes and Gonzalez [2015]). Thus, the course of activities during a construction process will influence RTC.

Usually, faults are made during a construction process, i.e. some activities are not correctly completed at the first attempt, but have to be redone. This phenomenon is called rework and is thought to be a major cause of schedule and cost overruns of construction projects. Therefore, the dependencies among activities are investigated with respect to RTC.

Browning and Eppinger [2002] developed a simulation model of a product design process in which rework occurs. The model is used to estimate design cost and duration probability density distributions. i.e. to estimate the expected interval for the cost and duration of a design process. Moreover, a criticality matrix is used to identify sources of rework.

Mommers [2015] modified the simulation model of Browning and Eppinger [2002], which has been implemented by Schuijbroek [2015] using the simulation languages Chi 3 Hofkamp [2015], such that it represents a product construction process in which rework occurs. The modified simulation algorithm is used to show the effect of rework on cost and duration distributions of a simplified lock construction process. The criticality matrix is used to identify activities which have a significant contribution to RTC.

8.1 Simplified lock construction

To illustrate the effect of rework using a highly simplified version of a detailed construction schedule of a recently realized lock. The construction schedule is divided into eleven aggregated activities. Each activity is characterized by a number of sub-activities. The eleven aggregated activities are:

1. **Soil preparation** consists of doing ground work, placing ground anchors and pouring underwater concrete.

2. **Cofferdam construction** consists of building the cofferdam ('bouwkuip') and draining the building area.

3. **Foundation construction** consists of doing iron work and pouring concrete for the lock foundations.
4. **Lock heads construction part I** consists of doing iron work and pouring concrete to build the lock heads ('sluishoofden').
5. **Lock heads construction part II** consists of finishing up the lock heads.
6. **Door hinges installation** consists of installing the door hinges.
7. **Mechanical components installation** consists of installing the mechanical components and cylinders which are needed to operate the doors.
8. **Lock chamber wall construction** consists of building the lock chamber ('kolk') walls.
9. **Hydraulics installation** consists of installing the hydraulics which power the doors and water leveling system.
10. **Doors installation** consists of placing and connecting the doors.
11. **Test** consists of testing the lock system.

8.2 Process, rework probability, and impact DSMs

The activities listed in the previous section are not independent. For example, activity 4 'Lock head construction part 1' requires that activity 3 'Foundation construction' is correctly finished before it can start. Figure 8.1 shows a process DSM (Eppinger and Browning [2012]) in which these dependencies are modeled. The sequence of activities along the diagonal denote the sequence in time in which the activities are done, i.e. it represent the construction schedule. An entry at position i, j indicates that activity j needs to be correctly completed before activity i may start. That is, one has to check and verify the correctness of activity j before one can proceed with activity i.

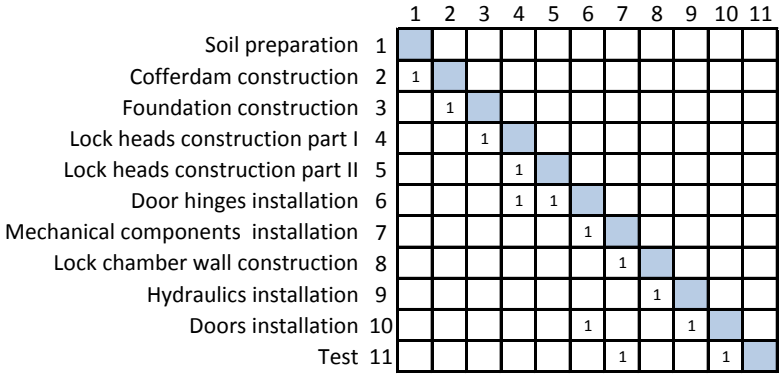


Figure 8.1: Simplified lock construction process DSM

However, there is a probability that activity i is not correctly completed and needs rework. Figure 8.2 shows the rework probability matrix M_p. Upper diagonal entries denote first order rework probabilities. That is, an upper-diagonal entry at position i, j denotes the probability that one first has to rework activity i before one can start activity j. Lower-diagonal entries denote second-order rework probabilities. That is, a lower-diagonal entry at position i, j denotes the probability that after applying rework to activity j, one has to apply rework to activity i as well. For example, Figure 8.2 shows that there is a probability of 0.10 that one has to rework activity 4, before one can

start activity 6. If one has to return to activity 4 before starting activity 6, there is a probability of 0.70 that one also has to rework activity 5 since it continued with the faulty results of activity 4.

Figure 8.3 shows the impact matrix M_I. An entry at position i, j denotes the fraction of work which has te be redone on activity i in case mistakes are detected prior to starting activity j. For instance, in the book shelve example only one of the two supports has te be reinstalled in order to level the shelve. As such, only a fraction of 0.5 of the original work has to be redone.

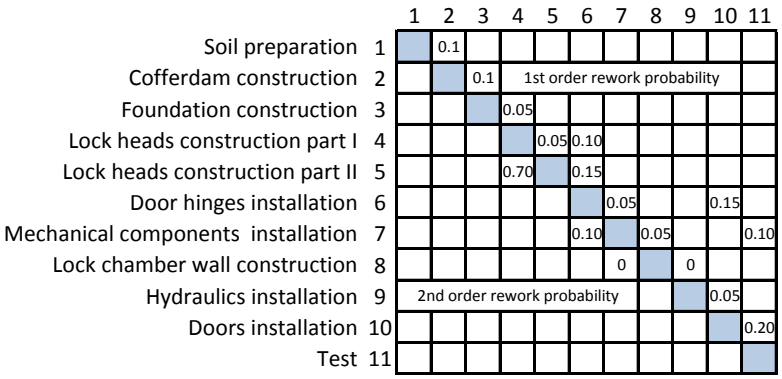


Figure 8.2: The rework probability matrix M_p.

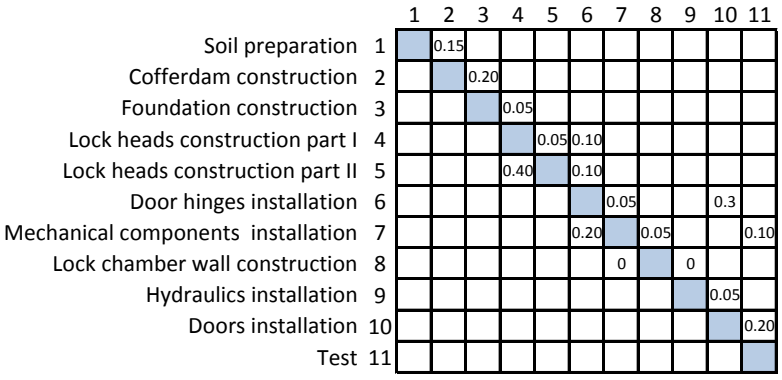


Figure 8.3: The impact matrix M_I.

8.3 Duration and costs distributions

The duration and cost of an activity are stochastic variables, i.e. they cannot be a priori predicted with 100% certainty. The values that a stochastic variable may take can be described by a probability density function. Table 8.1 shows estimated best case, most likely, and worst case values for the duration and cost of each activity. These values are not based on historical data. These values are used to fit gamma distributions which describe the stochastic nature of the durations and cost of each activity. The fitted distributions serve as input for the simulation model.

In real life a long duration is usually associated with a high costs. As such, Mommers [2015] has implemented a method to correlate the duration samples with the cost samples, i.e. to ensure if an activity has a long duration is also has a high cost and vice versa.

		Duaration [days]			Cost [kEuro]		
		BCV	MLV	WCV	BCV	MLV	WCV
Soil preparation	1	32	36	41	800	830	890
Cofferdam construction	2	33	35	39	820	850	890
Foundation construction	3	32	34	38	785	810	850
Lock heads construction part I	4	27	29	33	750	780	815
Lock heads construction part II	5	29	31	35	745	760	795
Installing door hinges	6	26	28	33	685	710	750
Installing mechanical components	7	28	29	32	1010	1040	1100
Lock chamber wall construction	8	30	31	33	1100	1135	1190
Installing the hydraulics	9	25	27	32	900	915	1010
Installing the doors	10	23	26	31	910	955	1010
Testing	11	27	29	36	870	930	1005

Table 8.1: Best case, most likely and worst case value estimations for the duration and cost of each construction activity.

8.4 Simulation algorithm

The simulation algorithm requires the process DSM, the rework probability matrix M_{p}, the impact matrix M_{I}, and the duration and cost distributions as input. The algorithm comprises the following steps:

1. Check if activity j requires work.
2. Check which activities $j < i$ need to be correctly finished before activity j can start.
3. Check whether first and second order rework is required on any of the activities $j < i$.
   (a) If required, apply rework and proceed in time.
4. Sample a cost and duration for activity j.
5. Start activity j, and proceed in time.

These steps are repeated for $j = 1 \ldots N$, where N is the number of activities the process consists of.

8.5 Dependency criticality

To find high risk activity dependencies, i.e. dependencies which may cause costly and lengthy rework, Browning and Eppinger [2002] introduced the criticality matrix M_{C}. Upper diagonal entries of M_{C} denote first order rework criticality and are given by Equation (8.1):

$$M_{\mathrm{C}}(i,j) = M_{\mathrm{p}}(i,j) \cdot M_{\mathrm{I}}(i,j) \cdot MLV(i) \text{ for } i < j \tag{8.1}$$

where $M_{\mathrm{p}}(i,j)$ is the probability that rework is required on activity i before activity j may start, $M_{\mathrm{I}}(i,j)$ is the fraction of rework that is required on activity i before activity j may start, and $MLV(i)$ is the most likely cost or duration of activity i. As such, it is possible to calculate a criticality matrix with respect to cost and with respect to duration.

Lower diagonal entries of M_{C} denote second order rework criticality and are given by Equation (8.2):

$$M_{\mathrm{C}}(i,j) = M_{\mathrm{p}}(i,j) \cdot M_{\mathrm{I}}(i,j) \cdot MLV(i) \cdot \sum_{k=j}^{n} (M_{\mathrm{p}}(j,k) \cdot M_{\mathrm{I}}(j,k)) \text{ for } i > j \tag{8.2}$$

where $M_{\mathrm{p}}(i,j)$ is the probability of second order rework on activity i after first order rework on j, $M_{\mathrm{I}}(i,j)$ is the fraction of required rework on activity i, MLV(i) is the most likely cost/duration value of activity i, $M_{\mathrm{p}}(j,k)$ is the probability that activity j requires first order before activity k can start, $M_{\mathrm{I}}(j,k)$ is the fraction of required rework.

8.6 Results

Figure 8.4 and Figure 8.5 show the total process duration and cost distributions, which result from running the simulation algorithm a thousand times. The blue line is the fitted probability density function, the green line is the cumulative density function which is obtained by taking the integral of the probability density function.

Figure 8.4 shows two duration distributions, the top graphs shows the resulting duration distribution if one does not account for rework, and the bottom graph shows the resulting duration distribution if one does account for rework. In the top graph approximately 95% of all simulation runs has a duration shorter than 350 days. In the bottom graph approximately 65% of all simulation runs has a duration shorter than 350 days. The 'fat tail' of the probability density function in the bottom graph indicates that rework can only make 'things worse' and increases the variation in the construction process duration. As a result, the chance that one observes a significantly higher duration, compared to the expected value, increases.

Figure 8.5 shows two cost distributions, the top graph shows the resulting cost distribution if one does not account for rework, and the bottom graph shows the resulting cost distribution if one does account for rework. In the top graph approximately 95% of all simulations runs has a cost lower than 9.85M. In the bottom graph approximately 50% of simulation runs has a cost lower than 9.85M. Again the 'fat tail' is observed, which makes it more likely that the construction cost will be significantly higher than the average value.

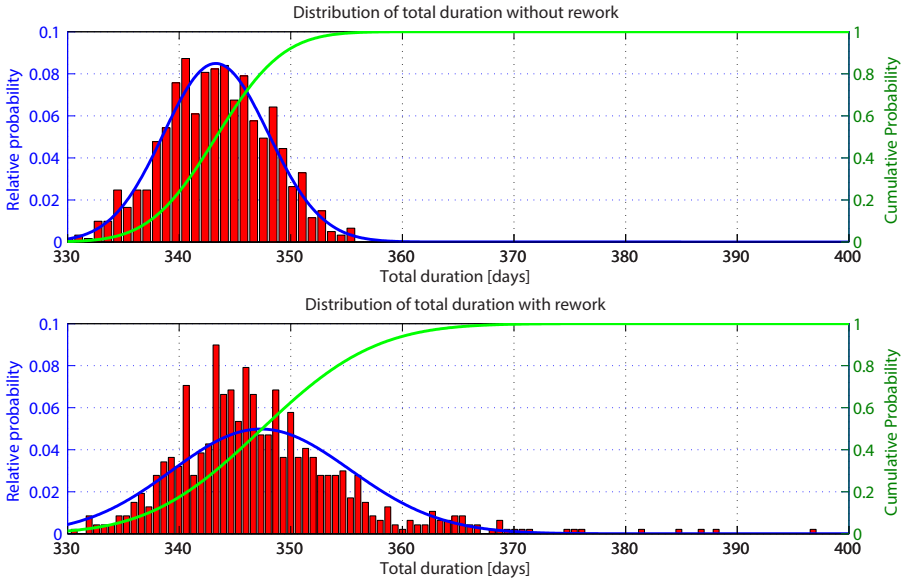


Figure 8.4: Resulting simulated duration distributions, without (top) and with rework (bottom).

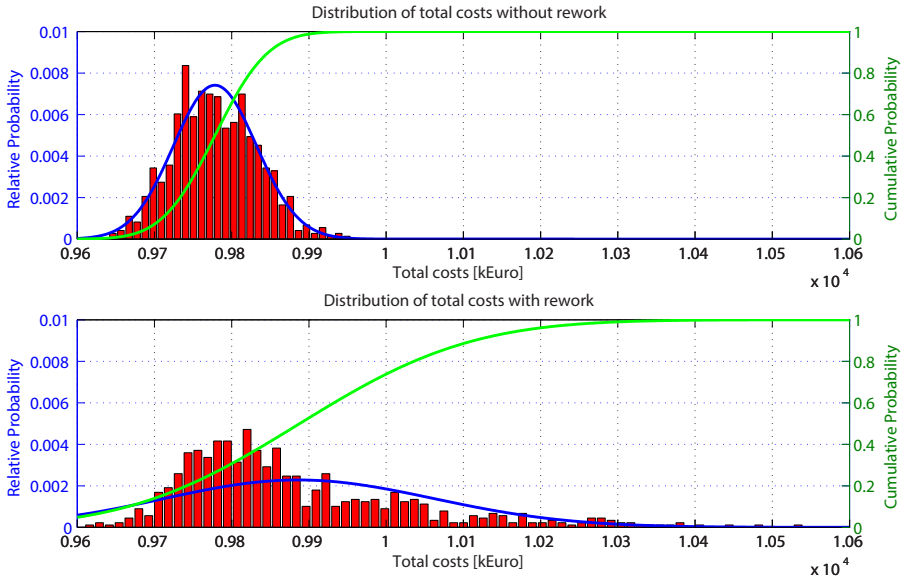


Figure 8.5: Resulting simulated cost distributions, without (top) and with rework (bottom).

The duration criticality matrix, shown in Figure 8.6, indicates that the dependencies between activities 6 and 10, and 10 and 11 are the most likely to cause lengthy rework. During activity 6 the door hinges are installed, during activity 10 the doors are placed, activity 11 consists of testing the lock system. Activities 6 and 10 have a relatively high risk of being completed incorrectly. Moreover, a relatively high fraction of rework is required if these activities are completed incorrectly. Therefore, these dependencies are denoted as high risk dependencies.

The risk on rework could be reduced by for instance installing dummy doors during activity 6 to ensure that the doors which are to be installed during activity 10 will fit and operate correctly. If one assumes that such a precaution measurement lowers the chance of rework by half, the activity dependency criticality is reduced as well, as is shown in Figure 8.7. Re-simulating the improved construction process yields the bottom duration distribution shown in Figure 8.8. The top graph shows the duration distribution of the original process, i.e. without the additional precaution measurement. Note that the length of the tail of the bottom distribution graph is shorter and leaner than the tail of the top graph. As such, the precaution measurement has reduced the chance of significantly overrunning the average duration.

Figure 8.7 shows that, as a result of the precaution measurement, the dependencies between activities 6, 10 and 11 are no longer the most likely sources of lengthy rework. The dependencies between activities 1 and 2, and 2 and 3 are now ranked as most critical. As such, one could also implement precaution measurements to lower the chance of rework within these activities. Iteratively adding precaution measurements can reduce the risk of schedule and cost overruns of the complete process.

8.7 Conclusion

The method of Browning [2001] has been effectively implemented by Schuijbroek [2015] and modified by Mommers [2015] to simulate the cost and duration of a lock construction process. By use of a simplified lock construction process example it is shown that rework may have a significant impact on the cost and duration of a construction process. Since rework can only make 'things' worse it may increase the risk of cost and schedule overruns. However, by studying the criticality

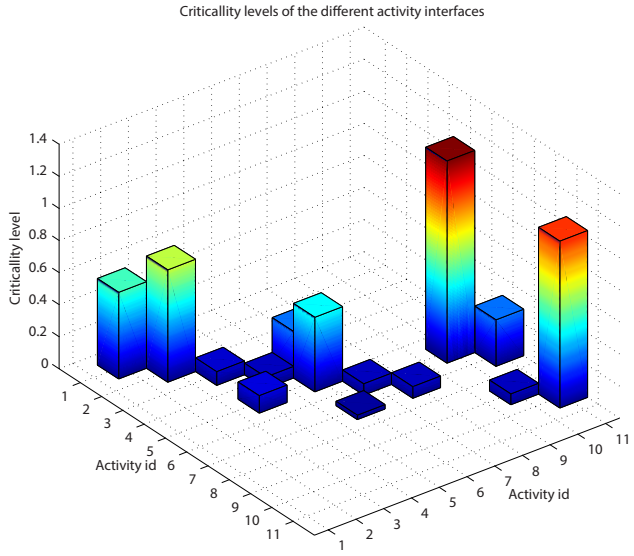


Figure 8.6: Cost critically matrix, indicating high risk activity dependencies.

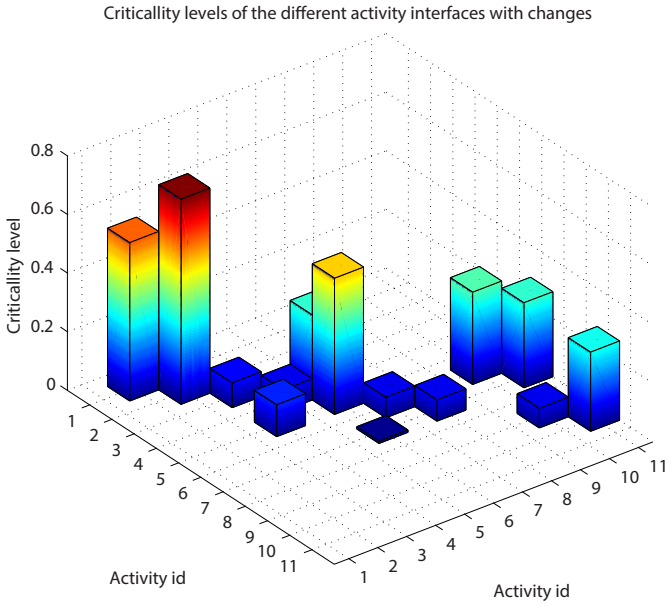


Figure 8.7: Improved cost critically matrix, indicating high risk activity dependencies.

matrix one can determine the sources of rework within the construction process and take precautionary measures to lower the probability of rework which in turn reduces the chance of schedule and budget overruns.

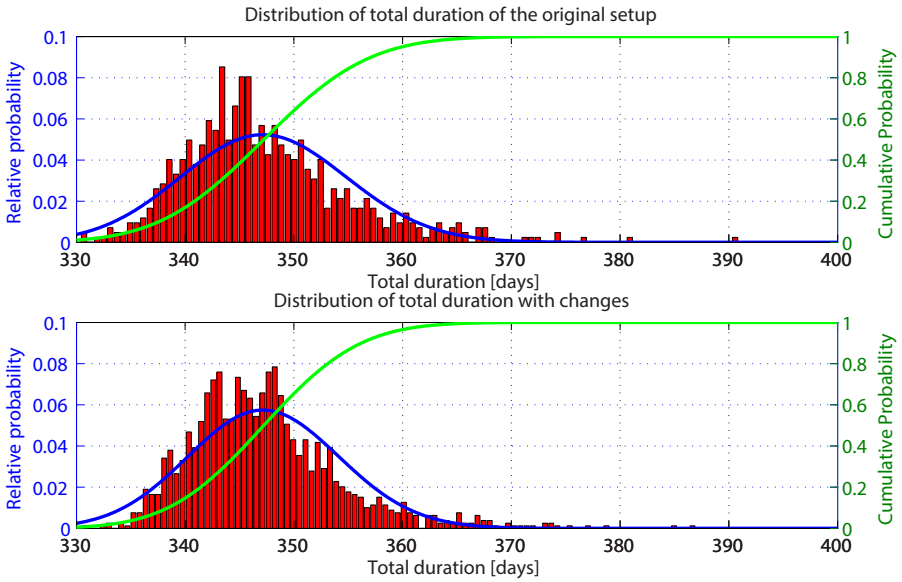


Figure 8.8: Simulated duration distributions without (top) and with (bottom) reduced rework probabilities.

Chapter 9

Conclusions and recommendations

This study investigates possibilities for modularization and standardization of locks by means of a design structure matrix approach. Through modularization and standardization, it is aimed to increase the reliability and availability (RA), reduce the life cycle costs (LCC) and decrease the risk of exceeding planned constructions time and costs (RTC) for a series of 52 locks, which are due for renewal before the year 2040.

Public lock tenders issued by RWS consist of functional requirements specified for conceptual components, i.e. components of which the physical form has yet to be determined. Contractors are responsible for the embodiment design, the detailed design, and the manufacturing of the lock. During each step the contractors have to make thousands of decisions which may all influence RA, LCC and RTC. The challenge for RWS is to control RA, LCC and RTC via the specification of functional requirements. Hence, the desire of MWW to apply modularization and standardization.

A modularization strategy specified from a functional and conceptual design perspective would best fit the needs and desires of MWW, since it operates in an 'engineer to order' industry. In fact, current public lock tenders already describe a conceptual design of a lock.

Specifying that two conceptual components are subject to the same functional requirement, actually implies that a functional dependency between these two conceptual components is possibly present. As a result, the contractors have to ensure that the physical realizations of the two components are such that together they deliver the desired functionality. So through the specification of functional requirements for conceptual components RWS influences the physical design of a lock. Therefore, a thorough understanding of the various function-component dependencies is essential for a proper and clear specification of a public tender.

The functional dependencies between conceptual components can be efficiently modeled and visualized using design structure matrix (DSM) techniques. The resulting image provides an abstract but analytically advantageous representation of the conceptual design of a lock, i.e. it shows which components have to work together to deliver the specified functional requirements.

It is advised to develop a function-component lock family platform. The function-component family platform consists of a 'common core' of functions and conceptual components which are required in every lock, and optional modules which consist of functions and components that are not always required. The functions/components in the common core and the optional modules are either fully standardized, partially standardized, or non-standardized.

RWS can increase their understanding of function-component dependencies, leverage development costs over multiple locks, reduce the number of required spare parts, lower renovation costs, and increase their influence of RWS on RA, LCC and RTC of locks by creating a 'nuanced' design freeze by means of the function-component platform.

This study shows that by modeling and analyzing lock function-component dependencies, using design structure matrix (DSM) based techniques, it is possible to identify the 'common core', identify optional modules, and identify functions/components which may be suitable candidates for (partial) standardization. For those functions/components which are denoted as possible candidates for standardization, it is required to:

- Investigate the *dependency on location specific parameters*, i.e. the design of a component maybe dependent on many location specific factors. As such, it might not be possible to standardize a component.
- Investigate the *function ranges*, e.g. one may find pumps in locks with a capacity ranging from $5m^3/s$ to 15 $15m^3/s$. As such, one may need several (standardized) variants of the same component.
- Investigated the *innovation rate*, i.e. the rate at which components change over time.

in order to decide whether it is beneficial to standardize a function/component.

Analysis of the function-components dependency structures of a series of seven locks shows that the functions 'F6.1-2 Openen/ sluiten sluisdeuren' and 'F7.5 Monitoren sluis systeem' and components 'Bewegingswerken' and 'Sluisdeur' may have a significant impact on RA, and may be suitable candidates for standardization. However, for a more complete lock family analysis it is essential to identify and model all dependencies between all functions and all components from 'Bomenschutsluis' at a sufficiently detailed decomposition level.

The Lock Eefde DSM study of Dijkstra [2015] showed that most of the complexity of a lock is due to energy and information dependencies. Moreover, a lock can be divided into distinct modules ('bouwdelen') that consist of several components. A particular interesting module is the so called 'bus' module. Bus components typically serve as coordination hubs within a system and therefore have many dependencies with many different components. As such, bus components require special attention during design and manufacturing to ensure proper functioning of the complete lock. Bus components can benefit from (partial) standardization. In the DSM study lock Eefde, the 'Besturingssysteem', the 'Laagspanningsinstallatie' and the 'Noodstroomvoorziening' are part of the bus.

Dijkstra also analyzed the dependency structure of lock Eefde with respect to availability and reliability using fault tree data based upon the current condition of lock Eefde. Dijkstra identified dependencies between 'Bewegingswerken' and 'Hefdeuren' and 'Bewegingswerken' and 'Sluishoofdconstructie' as being high impact dependencies with regards to availability and reliability. Therefore, these components seem to be suitable candidates for standardization. Note in this respect that the German Bundesantstalt fur Wasserbau applies full standardization of the all moving parts of a lock for a family of 26 locks.

Finally, Schuijbroek [2015] and Mommers [2015] investigated the effect of rework on the cost and duration of a lock construction project. Not all activities which comprise a construction project are correctly completed at the first attempt but have to be reworked.

The method of Browning [2001] has been effectively implemented by Schuijbroek [2015] and modified by Mommers [2015] to simulate the cost and duration of a lock construction process. Using a simplified lock construction process example it is shown that rework may have a significant impact on the cost and duration of a construction process. Since rework can only make 'things' worse it increases the risk of cost and schedule overruns. By studying the criticality matrix one can identify the critical sources of rework within the construction process and take precautionary measures to lower the probability of rework which in turn reduces the chance of schedule and budget overruns.

Bibliography

AA Alex Alblas and JC Hans Wortmann. Function-technology platforms improve efficiency in high-tech equipment manufacturing: A case study in complex products and systems (CoPS). *International Journal of Operations & Production Management*, 34(4):447–476, 2014. URL http://www.emeraldinsight.com/journals.htm?articleid=17107748&show=abstract.

Alex Alblas, Hans Wortmann, and others. The need for function platforms in engineer-to-order industries. In *DS 58-3: Proceedings of ICED 09, the 17th International Conference on Engineering Design, Vol. 3, Design Organization and Management, Palo Alto, CA, USA, 24.-27.08. 2009*, 2009. URL http://www.designsociety.org/publication/28614/the_need_for_function_platforms_in_engineer-to-order_industries.

Ricardo Antunes and Vicente Gonzalez. A Production Model for Construction: A Theoretical Framework. *Buildings*, 5(1):209–228, March 2015. ISSN 2075-5309. doi: 10.3390/buildings5010209. URL http://www.mdpi.com/2075-5309/5/1/209/.

A. Terry Bahill, L. William, and others. A tutorial on quality function deployment. 1993. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.369.8326.

Alessandro Birolini. *Reliability engineering*, volume 5. Springer, 2007. URL http://link.springer.com/content/pdf/10.1007/978-3-642-39535-2.pdf.

Timothy K. Brady. Utilization of dependency structure matrix analysis to assess complex project designs. In *ASME 2002 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pages 231–240. American Society of Mechanical Engineers, 2002. URL http://proceedings.asmedigitalcollection.asme.org/proceeding.aspx?articleid=1571498.

S. P. Brooks and B. J. T. Morgan. Optimization using simulated annealing. *The Statistician*, pages 241–257, 1995. URL http://www.jstor.org/stable/2348448.

T.R. Browning and S.D. Eppinger. Modeling impacts of process architecture on cost and schedule risk in product development. *IEEE Transactions on Engineering Management*, 49(4):428–442, November 2002. ISSN 0018-9391. doi: 10.1109/TEM.2002.806709.

Tyson R. Browning. Applying the design structure matrix to system decomposition and integration problems: a review and new directions. *Engineering Management, IEEE Transactions on*, 48(3):292–306, 2001. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=946528.

Michel-Alexandre Cardin, Gwendolyn L. Kolfschoten, Daniel D. Frey, Richard de Neufville, Olivier L. de Weck, and David M. Geltner. Empirical evaluation of procedures to generate flexibility in engineering systems and improve lifecycle performance. *Research in Engineering Design*, 24(3):277–295, July 2013. ISSN 0934-9839, 1435-6066. doi: 10.1007/s00163-012-0145-x. URL http://link.springer.com/article/10.1007/s00163-012-0145-x.

Michel Dijkstra. Reliability and avialability of lock component interactions. Masters' thesis, Eindhoven University of Technology, 2015.

Marco Dorigo, Mauro Birattari, and Thomas Stutzle. Ant colony optimization. *Computational Intelligence Magazine, IEEE*, 1(4):28–39, 2006. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4129846.

Steven D. Eppinger and Tyson R. Browning. *Design structure matrix methods and applications*. MIT press, 2012. URL http://books.google.nl/books?hl=nl&lr=&id=MPQoumoGXHYC&oi=fnd&pg=PR7&dq=stewart+design+structure+matrix&ots=ttwoef-9g3&sig=LUYqnprbMTLYzio-JisSxmoYAqU.

Joseph P. Ficalora and Louis Cohen. *Quality function deployment and six sigma: A QFD handbook*. Pearson Education, 2009. URL https://books.google.nl/books?hl=nl&lr=&id=K9sV3DyMQvAC&oi=fnd&pg=PT3&dq=Quality+function+deployment+and+6-sigma&ots=qp9L3714ni&sig=p9-6c6WDpRIZLsTIxzzAsJMraXo.

Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3-5):75–174, February 2010. ISSN 03701573. doi: 10.1016/j.physrep.2009.11.002. URL http://linkinghub.elsevier.com/retrieve/pii/S0370157309002841.

Fred Glover. Tabu search-part I. *ORSA Journal on computing*, 1(3):190–206, 1989. URL http://pubsonline.informs.org/doi/abs/10.1287/ijoc.1.3.190.

Fred Glover. Tabu searchâĂŤpart II. *ORSA Journal on computing*, 2(1):4–32, 1990. URL http://pubsonline.informs.org/doi/abs/10.1287/ijoc.2.1.4.

Carlos Gorbea, Tobias Spielmannleitner, Udo Lindemann, Ernst Fricke, and others. Analysis of hybrid vehicle architectures using multiple domain matrices. In *DSM 2008: Proceedings of the 10th International DSM Conference, Stockholm, Sweden, 11.-12.11. 2008*, 2008. URL http://www.designsociety.org/download-publication/27453/analysis_of_hybrid_vehicle_architectures_using_multiple_domain_matrices.

Cor PM Govers. What and how about quality function deployment (QFD). *International journal of production economics*, 46:575–585, 1996. URL http://www.sciencedirect.com/science/article/pii/0925527395001131.

A. T. Hofkamp. Chi 3 Reference Manual, 2015. URL http://chi.se.wtb.tue.nl/.

Jianxin Roger Jiao, Lianfeng Zhang, and Shaligram Pokharel. Process platform and production configuration for product families. In *Product Platform and Product Family Design*, pages 377–402. Springer, 2006. URL http://link.springer.com/chapter/10.1007/0-387-29197-0_16.

Jianxin (Roger) Jiao, Timothy W. Simpson, and Zahed Siddique. Product family design and platform-based product development: a state-of-the-art review. *Journal of Intelligent Manufacturing*, 18(1):5, February 2007. ISSN 09565515. doi: http://dx.doi.org/10.1007/s10845-007-0003-2. URL http://search.proquest.com/docview/200524355/abstract?accountid=27128.

Darren A. Keese, Andrew H. Tilstra, Carolyn C. Seepersad, and Kristin L. Wood. Empirically-derived principles for designing products with flexibility for future evolution. In *ASME 2007 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pages 483–498. American Society of Mechanical Engineers, 2007. URL http://proceedings.asmedigitalcollection.asme.org/proceeding.aspx?articleid=1604225.

Maik S Maurer. *Structural awareness in complex product design*. 2007. ISBN 3-89963-632-5 978-3-89963-632-1. URL http://mediatum2.ub.tum.de/doc/622288/622288.pdf.

Christoph Meier, Ali A. Yassine, and Tyson R. Browning. Design process sequencing with competent genetic algorithms. *Journal of Mechanical Design*, 129(6):566–585, 2007. URL http://computingengineering.asmedigitalcollection.asme.org/article.aspx?articleid=1449386.

Robin Mommers. The influence of rework on cost and schedule risks. Final Bachelor Project, Eindhoven University of Technology, 2015.

M.H. Nagel and T. Tomiyama. Intelligent sustainable manufacturing systems, management of the linkage between sustainability and intelligence - an overview. In *2004 IEEE International Conference on Systems, Man and Cybernetics*, volume 5, pages 4183–4188 vol.5, October 2004. doi: 10.1109/ICSMC.2004.1401187.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: Bringing order to the web. 1999. URL http://ilpubs.stanford.edu:8090/422.

Gerhard Pahl, Wolfgang Beitz, JÃűrg Feldhusen, and Karl-Heinrich Grote. *Engineering design: a systematic approach*, volume 157. Springer, 2007. URL http://books.google.nl/books?hl=nl&lr=&id=57aWTCE3gE0C&oi=fnd&pg=PA1&dq=Engineering+Design+Pahl&ots=UcX3VW0USc&sig=IsPR_wv1Qvy3TTmBPaY9booXt_Q.

Robert Phaal. *Roadmapping bibliography*. 2009. URL http://www.ifm.eng.cam.ac.uk/uploads/Research/CTM/Roadmapping/Roadmapping_Bibliography_Phaal.pdf.

Colin R. Reeves. A genetic algorithm for flowshop sequencing. *Computers & operations research*, 22(1):5–13, 1995. URL http://www.sciencedirect.com/science/article/pii/0305054893E0014K.

Relatics. Relatics - Voor het beheren van uw projectinformatie, 2003. URL http://www.relatics.com/nl/.

P. J. J. Renders and J. E. Rooda. Methodisch ontwerpen van fabricagesystemen. 2004. URL http://se.wtb.tue.nl/documentation/lecnotes/movis/movis_2004.pdf.

Satu Elisa Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27–64, August 2007. ISSN 15740137. doi: 10.1016/j.cosrev.2007.05.001. URL http://linkinghub.elsevier.com/retrieve/pii/S1574013707000020.

Tiemen J.L Schuijbroek. The impact of process structure on cost and duration. Final Bachelor Project 0745291, Eindhoven University of Technology, January 2015.

D.V. Steward. The design structure system: A method for managing the design of complex systems. *IEEE Transactions on Engineering Management*, EM-28(3):71–74, August 1981. ISSN 0018-9391. doi: 10.1109/TEM.1981.6448589.

Nam P. Suh. Axiomatic design theory for systems. *Research in engineering design*, 10(4):189–209, 1998. URL http://link.springer.com/article/10.1007/s001639870001.

Stijn Van Dongen. Graph Clustering Via a Discrete Uncoupling Process. *SIAM Journal on Matrix Analysis and Applications*, 30(1):121–141, January 2008. ISSN 0895-4798, 1095-7162. doi: 10.1137/040608635. URL http://epubs.siam.org/doi/abs/10.1137/040608635.