UNIVERSITY OF TWENTE.

Applied product lifecycle management for scaling development

an information management methodology

Bram van Munster

Industrial Design Engineering

Management of Product Development

Faculty of Engineering Technology

University of Twente

July 11, 2022

DPM 1949

A thesis submitted in partial fulfilment of the requirements for the degree of M.Sc. in Industrial Design Engineering.

Examination board

Eric Lutters	Chairman
Marijn Zwier	Supervisor
Marcus Pereira Pessôa	External member
Stef Boerrigter	Mentor from company
Ronald Gorter	Mentor from company

Wavin Technology & Innovation
Rollepaal 20
7701 BS Dedemsvaart
www.wavin.com

UNIVERSITY
OF TWENTE.

University of Twente
Drienerlolaan 5
7522 NB Enschede
www.utwente.nl

First edition, June 24, 2022

Acknowledgements

I would like to thank Marijn Zwier for the academic supervision of my master's assignment. Furthermore, I want to thank Stef Boerrigter and Ronald Gorter for their valuable guidance and input at Wavin Technology & Innovation. Finally, I want to express my gratitude to my parents and my girlfriend Iris Iemenschot for their continuous support throughout my studies.

Abstract

When scaling up the development of a Product-Service System (PSS), involving many stakeholders, and collaborating with multiple development partners, managing product development information is a challenge. The combination of mechanical design, electronics design, firmware development, and software development plays a key role in this information management difficulty. Product Lifecycle Management (PLM), when combined with concepts from other fields such as Quality and Configuration Management can help in tackling this issue. However, there exists little guidance in realizing PLM in product development. To this aim, a PLM methodology was developed for Wavin Technology & Innovation (T&I). This methodology contains methods and prescriptions to manage the information of a PSS across its lifecycle, bound by central principles. Furthermore, a PSS development metamodel is included, structuring the information types and their traceability. The methodology is based on Wavin T&I product development through the combination of existing tools and implemented using an interactive handbook in the form of a website. The result is an improved shared understanding of the product definition and plans across project stakeholders. Overall, the methodology provides a stepping stone in PLM application and can be adapted to other companies scaling up their development efforts.

Contents

List of Figures

List of Tables

List of Abbreviations

1

Introduction

When developing complex products or Product-Service Systems (PSSs), consisting of a mechanical design, hardware design, firmware and software, the management of information such as specifications and decisions is a significant challenge. Wavin is facing this challenge in the development of the next generation of their smart underfloor heating controller. This research investigates the details of this challenge, what topics and concepts it pertains to, and how to tackle it academically and practically. The resulting methodology can be applied to realize improved information management practices across product development.

This report is structured as follows. First, background information on Wavin and Wavin product development is provided. Then, the problem is specified. To place this problem into academic context, an explorative analysis is performed. The development approach for a solution to the problem is discussed in the next chapter. Subsequently, the resulting methodology is discussed in detail. Finally, the results are discussed, and conclusions are drawn.

2

Background

2.1 | Company profile

Wavin is a manufacturer of, among others, plastic pipes, wastewater drainage systems, stormwater management systems and indoor climate solutions for heating and cooling. Founded in 1955 in Zwolle, Wavin operates across the globe in countries in Europe, North America, Latin America, Asia, and Oceania. In 2012, Wavin was acquired by Orbia, a global company headquartered in Mexico. The headquarter of Wavin is located in Amsterdam and the R&D subsidiary – Wavin Technology & Innovation (T&I) – is located in Dedemsvaart. New innovations of Wavin include the Ventiza ventilation ducting system, the Tigris K5/M5 pressure test fitting, and AquaCell infiltration units. Wavin develops and sells products and services in the Business Units (BUs) Hot and Cold, Urban Climate Resilience, Indoor Climate Solutions (ICS), Soil and Waste, and Water Management across the various Operating Companies (OCs) globally. Wavin T&I develops products and services for all business units. The ICS business unit, to which this research pertains, primarily operates in Europe, and its developments are lead from Dedemsvaart in the Netherlands as well as Hammel in Denmark. Being only a decade old, the ICS business unit is one of the youngest in the organization. The organizational structure is such that the product management takes place at the BU, driving development projects at T&I, which are managed by the Project Management Office (PMO) in Dedemsvaart.

2.2 | Sentio

Sentio is a smart underfloor heating controller that forms the cornerstone of the ICS BU. It is their second-generation underfloor heating controller, superseding the successful AHC-9000. The main products in the Sentio line are the CCU-208 controller, the EU-A and EU-VFR extension units, the wireless and wired thermostats, the LCD-200 control touch screen, and the Smart Radiator Thermostat (SRT) (all but the extension units shown in figure 2.1 and 2.2).

Figure 2.1: From left to right: the Sentio CCU, thermostat, and LCD-200

Figure 2.2: Sentio SRT

The Central Control Unit (CCU) can control up to 16 Underfloor Heating (UFH) actuators when paired with the EU-A. It can control the inlet temperature and pump of the UFH manifold as well as various heating sources. Using the EU-VFR, up to four dehumidifiers can be controlled by Sentio as well. Via an Ethernet connection, the CCU communicates with the cloud, allowing control via a mobile app. The CCU communicates with the wired and wireless thermostat, as well as the SRT through a proprietary protocol named ROXi BUS. This protocol is developed and owned by Jablotron, a Czech company to which the development of Sentio is outsourced by Wavin. In Appendix A, a schematic overview of Sentio components and communication is provided.

To gain insights on the indoor climate controls market, a market analysis was conducted. This analysis consisted of the comparison of competitors' offerings based on the following criteria: price, zone control capabilities, smart home integration, wireless connection capabilities, HVAC system integration, cooling support, UFH support, humidity control, air quality control, and miscellaneous smart or climate control features. From the

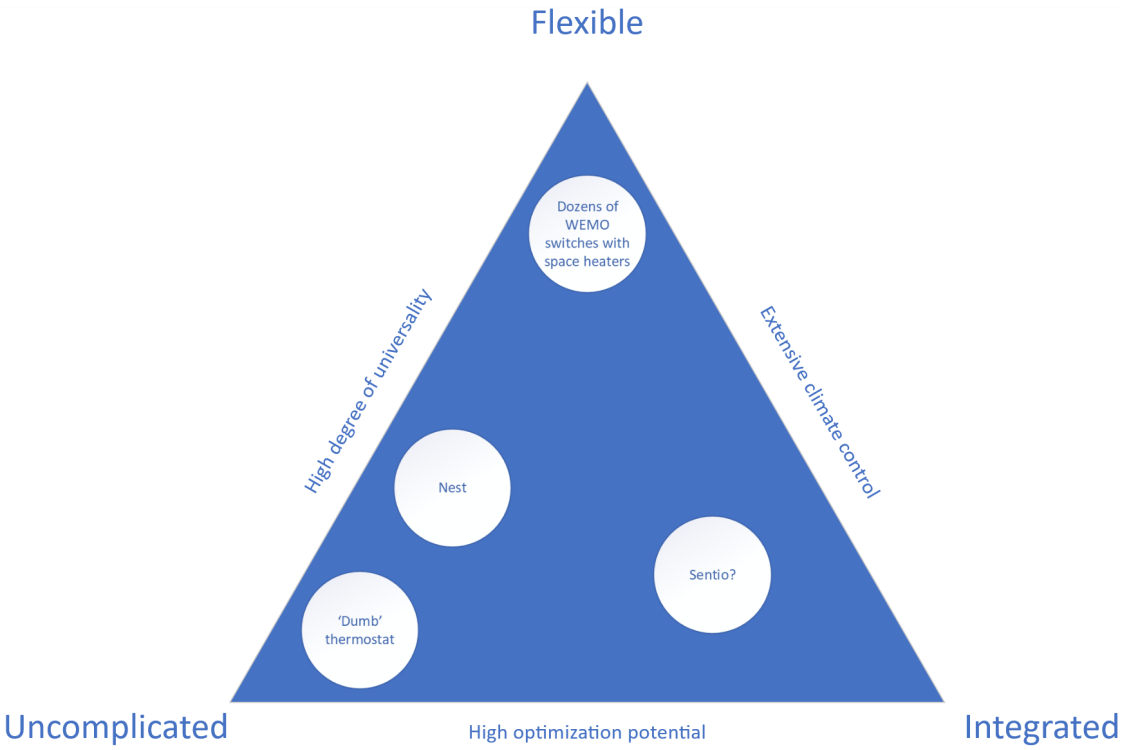


Figure 2.3: Ternary plot of indoor climate control characteristics

comparison of Sentio with 17 competitors, three main characteristics of indoor climate control systems were discovered: integration, simplicity, and flexibility, on which offerings can be mapped. The full comparison of offerings is provided in Appendix B. A suggestion for the placement of Sentio as well as a few alternatives are placed on a ternary plot of these characteristics in figure 2.3.

2.3 | Sentio 2.0

The consensus at Wavin is that the current CCU-208 is approaching its limits. New products such as the SRT and many firmware and software features have been added over the years increasing the firmware footprint. The new indoor climate solution intended to supersede Sentio is codenamed Sentio 2.0 within Wavin. The architecture is yet to be defined of this system, though there is an agreement on many of the features in general. In line with the intention to transition from product to service revenue from senior management, the definition of the next generation Sentio Product-Service System (PSS) must be developed.

2.4 | Project characteristics

In an early Sentio 2.0 brainstorm session on January the 25th, 2022, with application engineers from T&I, as well as representatives from the BU and various OCs, the ternary plot from figure 2.3 was presented for discussion. However, the everyday feature request tended to gain precedence over the definition of the climate control system as a whole, as they were otherwise thought to be lost, since there was no structured request process in place. This difficulty to maintain a holistic perspective due to the perceived urgency of everyday feature requests indicated a lack of facilitation for common understanding. This is a symptom of the novelty of the project: never before has a system definition be developed at Wavin T&I in conjunction with a current system.

Until now, Sentio has been developed by Jablotron on request; lists of requirements were negotiated, after which development and production was executed by Jablotron. For Sentio 2.0, execution of development will be managed by Wavin. Developments such as mechanical design, hardware design, and software development will not be executed by Wavin T&I, but they will be initiated, monitored, and integrated. This will reduce dependency on Jablotron as a development partner and open the door to additional development partnerships, potentially improving development capacity, flexibility, and time to market. Furthermore, Wavin wants to further consolidate their non-intrusiveness philosophy, based on the belief that end customers do not want to mindfully interact with their indoor climate. An elaboration on the non-intrusiveness philosophy can be found in Appendix C.

As part of a larger digitalization strategy, Wavin has initiated a department called the Digital Service Factory (DSF). Sentio is intended to be the first product to use the Wavin cloud developed by this DSF. Incorporating this fact, the development situation is illustrated in figure 2.4. Overall, the combination of the ICS business unit and the T&I ICS team at Wavin can be characterized as a start-up within an established company, experiencing the growth pains of scaling development as well as the rigidity of established development processes.

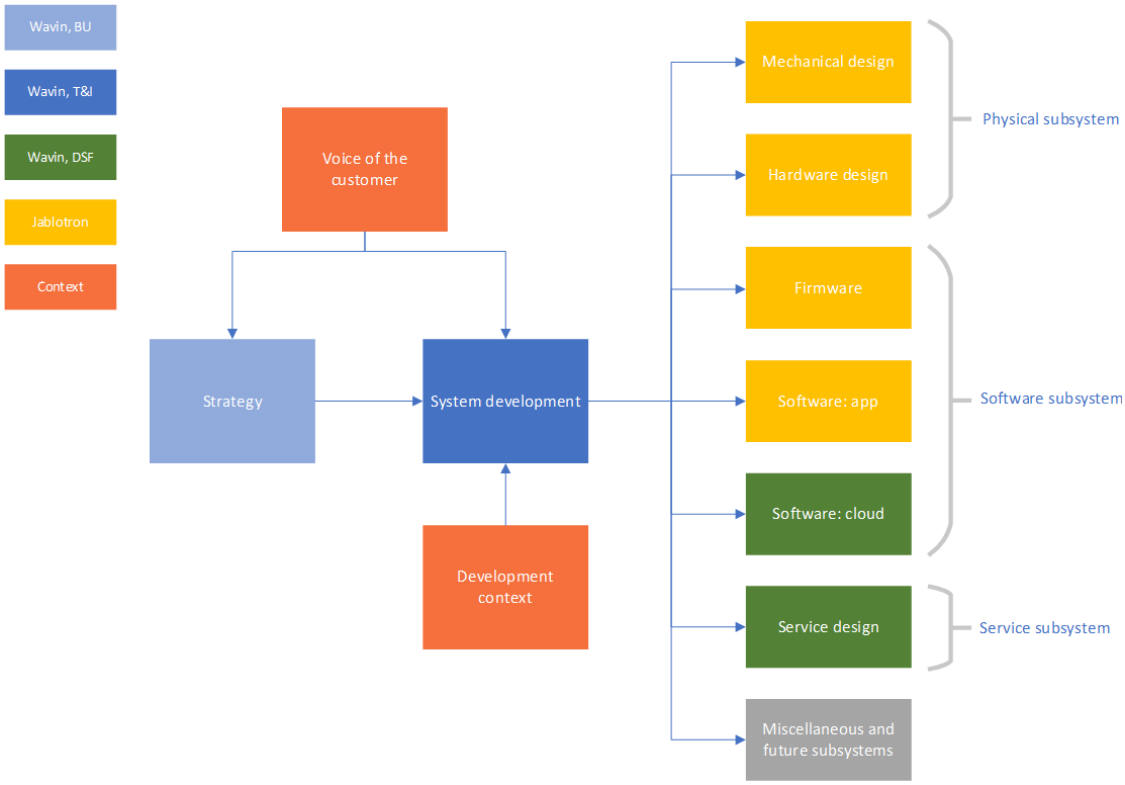


Figure 2.4: Illustration of the development situation

2.5 | Stakeholder analysis

In system development projects, many stakeholders are often involved. In this section, the stakeholders of the Sentio project are analyzed. First, an overview of the stakeholders is provided. Then, the characteristics of each stakeholder are discussed. Finally, they are placed in a power-interest matrix.

2.5.1 | Overview

As the ICS enterprise, comprising of the ICS business unit and the ICS team at Wavin T&I is still forming and growing, roles and responsibilities have not fully consolidated. Additionally, the interactions between stakeholders may change as the operation further develops. For this reason, it is important to maintain a clear overview of who is involved in the Sentio endeavor and how. Even though end users, wholesalers and installers are the customer (both primary and secondary) that will receive the developed system without having a direct influence on it, they should be incorporated as stakeholders in development. Retrospectively, after all, they are the most powerful, as they are the sole reason

for development. Incorporating the voice of customer in development as much as possible can reduce this retrospective component. Besides, as the business models assigned to Sentio may change in the future, installers may become partners in delivering value to end users, shifting their position from customer to stakeholder. In figure 2.5, a stakeholder map of the current state is shown.

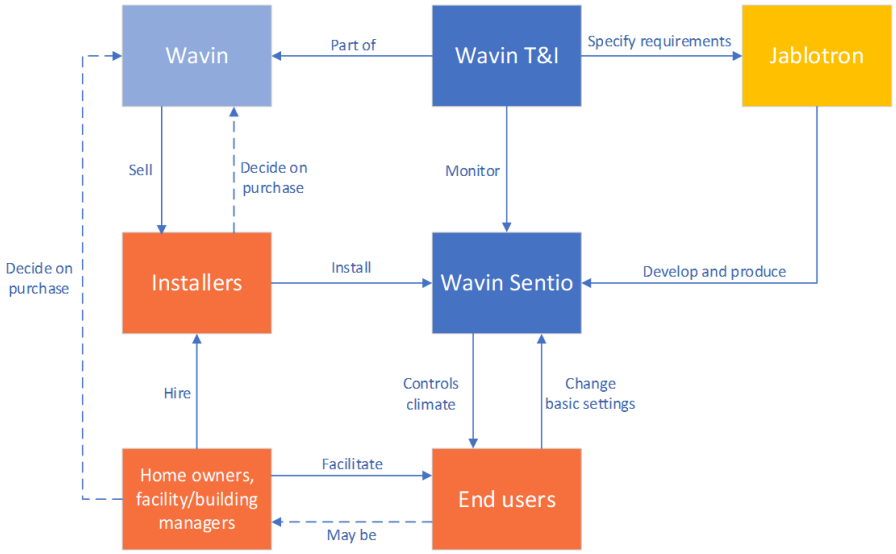


Figure 2.5: Stakeholder map, current state

The ambitions of Wavin to grow the ICS operation and deliver integrated solutions over commodities are realized in part by insourcing the development of a Wavin Cloud. This cloud is developed at the DSF mentioned in section 2.4. Sentio will be the first Wavin product to operate on this cloud. The DSF is added to the future state stakeholder map in figure 2.6.

2.5.2 | Stakeholder characterization

Stakeholders are grouped organizationally and characterized in this analysis. Where necessary, specific people or roles may be addressed as representative of this organizational group. These organizational groups are not geographically bound; most often, international collaboration is involved. Per stakeholder, a description of their power and interest is provided, as well their general interactions with Wavin T&I, such as the types of information that are exchanged. Note that, even though Wavin Sentio is included in the stakeholder map of figure 2.5 and 2.6, it is not (represented by) a person and, therefore, not a stakeholder. It serves a functional purpose to illustrate the relationships between the system and the stakeholders.

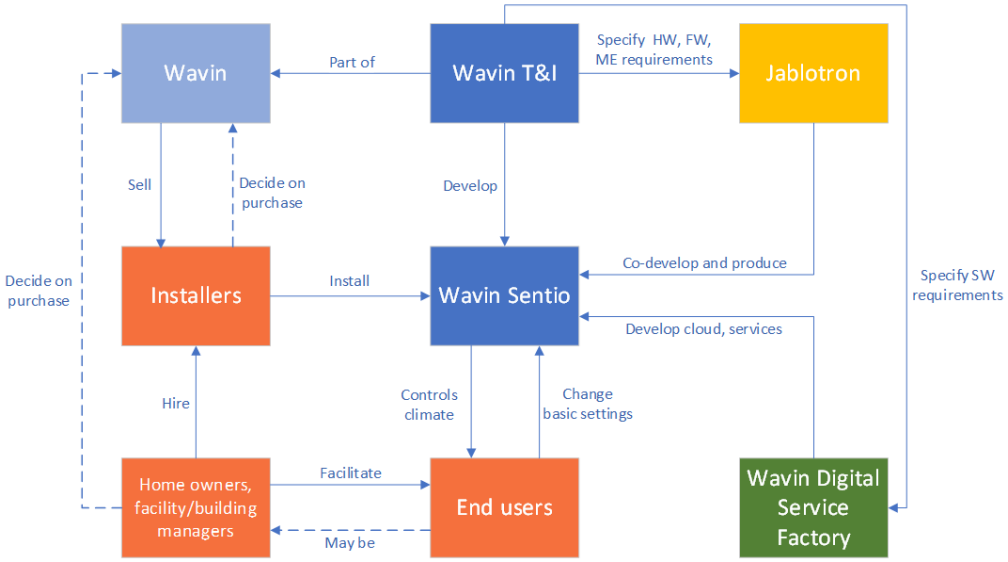


Figure 2.6: Stakeholder map, near future

2.5.2.1 | Wavin T&I

At Wavin T&I (more specifically, the Wavin T&I ICS team) in Dedemsvaart, all components of Sentio development coincide. Part of the T&I team working on Sentio works in Denmark: these are application and test engineers with extensive experience in underfloor heating in the Scandinavian market.

- Power: high
  As Sentio system development and integration occurs at Wavin T&I, its power is significant.
- Interest: high
  The ICS T&I department needs to fulfill the needs and wishes of senior management. Wavin as a whole, naturally, benefits from successful development projects.
- Interaction
  The Wavin T&I ICS team is the central viewpoint for the methodology under development. Its interaction with other stakeholders is discussed at their respective entries in this analysis.

2.5.2.2 | Wavin (Indoor Climate Solutions Business Unit)

Organizationally, the Wavin T&I ICS team develops the Sentio system for the Wavin ICS BU. This ICS BU is led by the Global BU Director working in Denmark, to which the Global

Product Manager Controls / Smart Home reports from the Netherlands. In practice, the Global Product Manager is quite narrowly involved with Sentio development for senior management due to the relative youth of the ICS operation. This Product Manager can be regarded as the representative of the senior management: the ICS business unit. Sales and marketing are performed by this BU.

- Power: high
  The ICS BU formally makes the final decisions about projects such as Sentio and is responsible for the strategy of Wavin's ICS endeavors.
- Interest: high
  Naturally, project successes contribute positively to the prosperity of the complete organization.
- Interaction
  - Information
    The business unit has access to all files, specifications, insights, et cetera. However, mostly project and product performance need to be regularly communicated with the BU. Still, the Product Manager can also be involved in the development of the system specification in order to ensure proper translation and application of the strategy provided by the business unit.
  - Characteristics
    Although the business unit is formally located in Amsterdam, BU members also work from Denmark, the United Kingdom, and more locations. For this reason, face to face meetings are relatively infrequent. Nevertheless, the digital infrastructure of Wavin ICT supports effective online meeting and collaboration.

2.5.2.3 | Jablotron

Jablotron is the Czech development and production partner that has worked with Wavin since the predecessor of Sentio: the AHC-9000. Currently, there are collaborations with the following specific subsidiaries of the Jablotron Group a.s.: JabloPCB s.r.o., Jablotron Controls, and Jablotron Cloud Services s.r.o. For the Calefa and domestic hot water (DHW) projects, Wavin T&I also collaborates with Logic Elements s.r.o. As the ICS operation matures, Wavin strives to insource more software and service-related components of Sentio, in order to deliver integrated solutions.

- Power: high
  Especially while most of technical development is performed by Jablotron, they

have the power to change the specification provided by Wavin T&I based on their capabilities and capacity.

- Interest: medium
  Wavin is an important development partner and customer of Jablotron, and project success strengthens this partnership. However, Jablotron is not dependent on this partnership.

- Interaction

  - Information
    Currently, specifications are agreed upon after discussion, after which Jablotron executes development and returns and produces (sub)system designs.
  - Characteristics
    Jablotron works with their own project management system named Phabricator, to which Wavin T&I has partial access. In there, T&I can view the status of the subprojects that Jablotron is working on.

2.5.2.4 | Wavin Digital Service Factory

One of the steps taken by the ICS business unit to deliver integrated solutions and increase servitization is to develop a Wavin Cloud in-house. This development will be part of the Wavin Digital Service Factory and Sentio will be the first product to make use of it. As this DSF is still 'under construction', communication and collaboration between it and the T&I ICS team is not yet defined. However, this provides the opportunity to work together efficiently from the start.

- Power: medium
  Depending on the strategy imposed by the ICS BU, services developed by the DSF could directly impose requirements on Sentio. However, especially while the DSF is still 'under construction', service and cloud implications for Sentio will be actively collaborated on.

- Interest: low
  The Digital Service Factory will develop and provide services for Wavin, as does T&I with system development in the case of Sentio. Sentio will be the first product to make use of the upcoming Wavin Cloud, but the DSF is not directly dependent on Sentio product development success.

- Interaction

- Information
  The initial information that will be exchanged between the DSF and T&I is the properties and constraints of Sentio's CCU, in order for the DSF to develop a suitable API to the Wavin Cloud. In this situation, T&I provides requirements (the limitations), and the DSF returns a specification (the API). In the future, the DSF can also impose requirements on Sentio development (e.g., minimum required capacity for cloud communication).
- Characteristics
  Interaction between T&I and the DSF can be determined as both teams develop. In this way, the degree of collaboration and the exchange of requirements and specifications can be refined and optimized over time.

2.5.2.5 | Installers

Installers are Wavin's direct customers for Sentio. They make their purchase at wholesalers and resell it to end users, adding an installation fee. The primary decision factors for purchase are cost price and ease of installation.

- Power: low
  Installers have no direct power in Sentio development decisions. As customers, they have a lot of power retrospectively, which can be partially mitigated by involving them in development. As mentioned before, this transitions them from customer to stakeholder.
- Interest: medium
  The interest of installers in Sentio is medium: installers benefit from Sentio development success in the form of end user satisfaction. Additionally, ease of installation allows them to work more efficiently, further increasing end user satisfaction through fewer installation faults and/or lower fees. Sentio being a unique, differentiating value proposition allows installers to sell a unique, differentiating value proposition to their customers. However, installer interest is limited due to competition in the market: installers are not dependent on Sentio.
- Interaction
  Installers experiment and learn by themselves how Sentio works. The Sentio technical handbook is the most comprehensive guide provided by Wavin for this purpose. Currently, there are only few videos available from Wavin, but there are plans to further expand the offering of online support tools for installers. From the Sentio webpage, the simple user manuals for the sensor and thermostat can be downloaded, as

well as the quick guide, which is roughly a simplified version of the technical handbook[1]. Support content varies per country: Dutch and English support pages, for example, have an FAQ section[2]. The Danish and French support page do not contain any support videos nor an FAQ section (yet). However, the Danish brochure does link to the YouTube channel of Wavin Denmark, which does contain Danish instruction videos for Sentio. In Scandinavia, Wavin also offers a service called 'All-Inclusive', where the right Sentio system components are prepackaged and the CCU is preprogrammed for convenience.

2.5.2.6 | Homeowners, facility managers, end users

Currently, Wavin is not planning on selling Sentio directly to end users. However, homeowners, facility managers, and end users can certainly express their preference for certain brands and products to their installers. In the case of new construction, the contractor can be regarded as the facility manager until construction is finished. Contractors, facility managers, homeowners and end users all make the decision for certain installers and the services and products they provide.

- Power: low
  Like with installers, but to an even lesser extent, homeowners, facility managers, and end users only have retrospective power in Sentio development through sales performance. If Wavin were to sell Sentio directly to end customers, this power would increase to the same level as the installers'.

- Interest: low
  End users benefit from the success of Sentio development in the form of product performance in the form of lower prices and/or increased comfort and convenience. However, due to competitive offers in the market, end users are not dependent on this success.

- Interaction
  End users can access the same public information as installers, which also includes instructions and manuals for basic settings that are relevant to the end user. Usually, complaints are directed to the installers, who, if they cannot resolve it themselves, relay them to Wavin. In short, communication to the end user occurs through documentation, manuals, et cetera. Communication from the end user to Wavin is limited. In the future, the amount of feedback can be improved by through user

[1] https://www.wavin.com/en-en/sentio/sentio-support
[2] https://www.wavin.com/nl-nl/sentio/sentio-support#Anchor%20faq

statistics, if or when more Sentio systems are connected to the internet and are logging user behavior.

2.5.3 | Power and interest

Mendelow [84] introduced the principle of aligning power and interest of stakeholders in order to direct attention to prioritized stakeholders. In figure 2.7, the power-interest matrix for the Sentio project is provided.

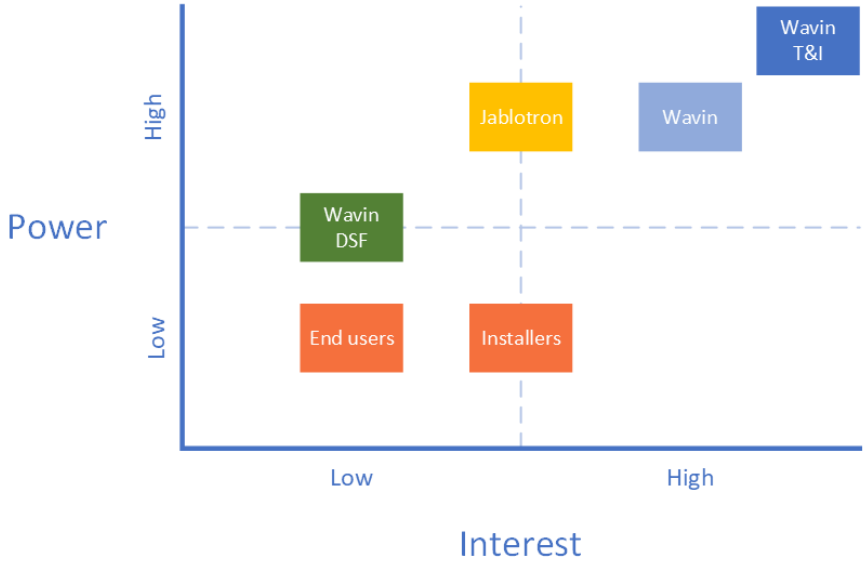


Figure 2.7: Power-interest matrix

3

Problem

As the current generation of the Sentio generation is reaching its limits, Wavin needs to gain insights on the management prerequisites for the development of the following generation(s). In this chapter, the problem statement is provided and elaborated through identified gaps in product development management.

3.1 | Problem statement

Due to the natural growth of the Sentio project and high degree of outsourcing, management of product development and its related information have been only sufficient until now. Currently, this results in the day-to-day overruling the discussion on larger questions such as the desired architecture and development strategy of the Sentio product line. In order to facilitate further growth of the ICS development efforts and the development of a new Sentio generation, the information management across the product lifecycle must be improved.

3.2 | Gaps

Gaps in the current information management practices have been identified and serve as a structure and goal for the development of a solution. With these each of these gaps, an explanation of its origin and guiding questions are provided. These guiding questions form the starting point of the explorative analysis as well as the proposed solution. In figure 3.1, the location of the gaps and their guiding questions are placed on the illustration of the development situation.

1. Requirements management
   Currently, lists of requirements only serve for agreements with Jablotron at Wavin.

They are not a full system specification, but they have been sufficient while most of development execution was performed by Jablotron. Additional features and updates have been requested mostly as they arose from various stakeholders, but they have never been combined into a single system specification. This is in line with the Agile methodology, avoiding so-called *Big Upfront Design*. However, this level of agility will not scale well as the project and its team grows, development shifts to Wavin, and more stakeholders get involved [5].

a) Voice of the customer – how can the voice of the customer be captured adequately and processed into system requirements?

b) Traceability – how can the implications of requirements be traced back to their origin?

c) Completeness – how can completeness of the requirements be ensured?

d) Consistency – how should conflicts between requirements be resolved?

e) Idea generation – how can internal ideas be filtered according to the PSS vision and translated into system requirements?

f) Triage – how can requirements for the system and subsystems be prioritized?

g) Configuration – how are multiple product architectures managed and how are requirements, wishes and issues assigned to the right architecture?

2. Change management
   Change management in system development is the extension of requirements management over time. A formalized system definition must stay relevant as both internal and external factors change. Until now, potential changes to Sentio have been stored in Excel sheets and have been implemented ad hoc. A defined change management approach can help the right requests be implemented over the louder requests.

   a) Requirements implications – how can a change process be aligned with the system development process?

   b) Elicitation – how and how often should changes be assessed? How should technological, industrial, societal, and commercial opportunities be identified and processed?

   c) Contextualization – which contextual factors must be monitored and how?

   d) Responsibility – how many people and who are necessary to translate changes to subsystem-level requirements?

e) Versioning – how should hardware, software, mechanical designs, documents, and baselines be versioned if system development is, for example, continuous?

3. Decision management
   Due to the relative youth of the ICS operation, decisions are not formally documented. The corporate transparency and flat hierarchy ensure that everyone's voice is heard. However, with the ICS team growing in the past years and the teams working decentrally, decision making may require more organization and more thorough documentation.

   a) Organization – how and by whom should requirements, plans and decisions be taken, disseminated, and aligned? Where does the project 'reside'?

   b) Documentation – how should decisions be documented? How and how often should documented decisions be reviewed and reconsidered (i.e., how does change management apply to general decisions)?

4. Knowledge management
   Similar to decision management, insights and their documentation are organized ad hoc. Finding reports and design decisions is currently time-consuming and not always successful. Applying knowledge management practices can improve the absorption, dissemination and application of insights attained from daily T&I research [92].

   a) Documentation – how should investigations, field trips, small reports, etc. that are not (yet) directly linked to a decision or requirement be documented? How should artefacts be linked to requirements and decisions?

   b) Definitions – (how) should a single source of truth of the system be approached?

5. Verification & validation strategy
   Sentio testing currently takes place at Jablotron, with certain tests being repeated or executed at Wavin T&I for verification. This process is not explicitly traceable to the list of requirements that was agreed upon before development. With development responsibility shifting to T&I, so does verification and validation responsibility.

   a) Verification – how and when can system quality (making the system right) be ensured for the whole system and its subsystems? I.e., how can the subsystem designs and implementations and their integration into the system architecture be verified robustly?

b) Validation – how and when can the value proposition (making the right system) be validated for the whole system and its subsystems? How can the PSS be validated against the vision and philosophy, as well as the senior management strategy?

6. Risk management
   A SWOT analysis has been performed early on in Sentio development and organizational risk management is the responsibility of senior management. Nevertheless, development risks should be monitored to adhere to time and budget constraints [105].

   a) Risk management – (how) can development risks be identified and acted upon effectively?

7. Vision and philosophy
   The ICS business unit steers its own course. The ICS team at T&I fulfills the development this set course requires. Certain philosophies and ambitions, such as system non-intrusiveness and the ambition to develop integrated (service-based) solutions are present and prominent, but not registered. If decisions are based on an undocumented philosophy or strategy, and the underlying philosophy or strategy changes, the decision becomes untraceably invalid.

   a) Senior management – how does the senior management strategy translate to a PSS vision?
   b) PSS philosophy – what are the values and definition of Sentio? How can they be (re)evaluated and applied in development?
   c) Development philosophy – what are the company values for Sentio development? How does this relate to collaboration?
   d) Development typology – should Sentio (versions) be developed linearly, iteratively, organically, continuously, or in any other fashion?

8. Process and methodology improvement
   If the Wavin ICS team at T&I adopts a development methodology, it is important to frequently reestablish whether the methodology is effective and fitting to the operational scale, and act accordingly.

   a) Monitoring – how, how often, and by whom should the development methodology and process(es) be monitored regarding performance and quality?

b) Process change management – how should the development process(es) and methodology be evaluated and changed over time?

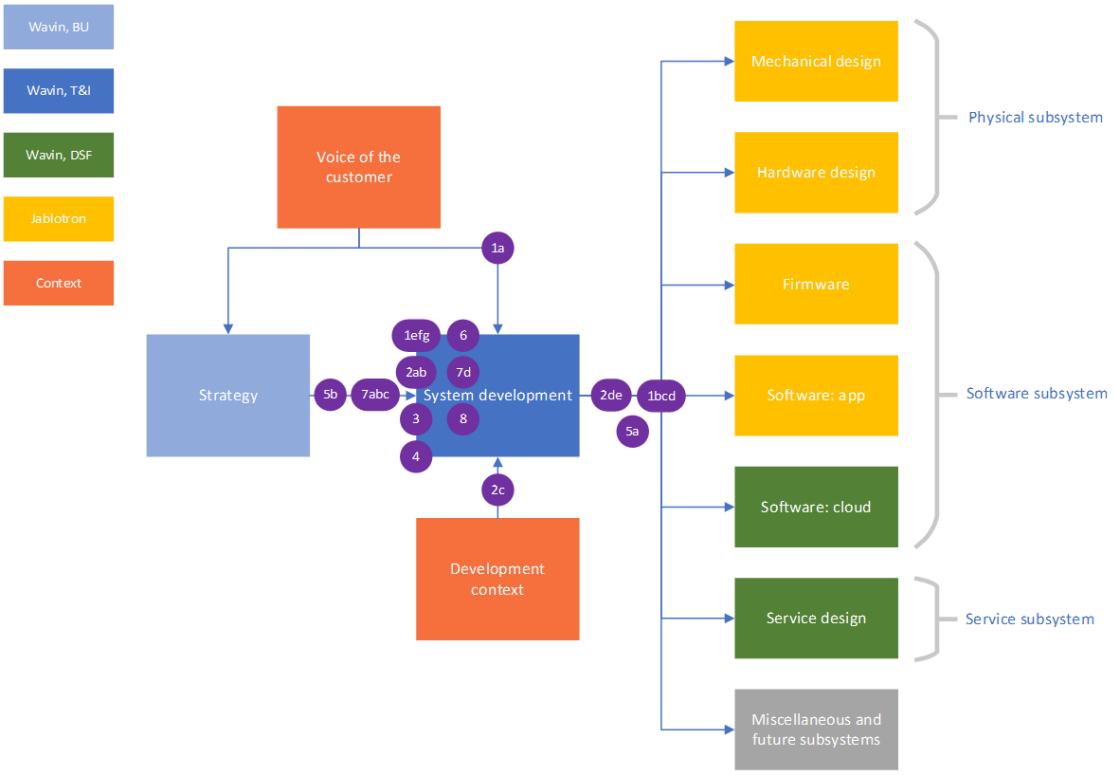


Figure 3.1: Locations of the information management gaps in the development situation

3.3 | Product lifecycle management

Product Lifecycle Management (PLM) is an evolving concept that integrates the management of all information on the lifecycle of a product, such as documents, BOMs, test results, change orders, manufacturing procedures, and so on [110, 123, 82]. Search trends show a significant increase in interest in PLM since 2014 [52]. Still, risks have been identified in the implementation of PLM systems, most notably divided into human-, process- and technology-related risks. The first involve factors such as limited user acceptance, lack of management support, and lack of technological skills. The second and third include risks such as lack of infrastructure and resources and lack of alignment between PLM technology, the development process, and business processes [121].

PLM maturity models generally assess strategy and policy, management and control, organization and processes, people and culture, and information technology [15]. A

higher PLM maturity can yield benefits in the areas of idea, requirements, change, and risk management, as well as product structuring, product program planning, project controlling, and quality controlling [115]. Multi-dimensional maturity assessment models provide significant benefits over one-dimensional roadmaps by providing a holistic view on PLM implementation [120]. Even though PLM assessments are inaccurate and provide little to no suggestion on how to proceed to improve maturity, they are useful for companies to determine their relative maturity with respect to other companies [101]. However, there is only use in the measurement of PLM maturity, if PLM efforts have been made in the past.

PLM systems are software suites organized around a central database that stores product information of the whole lifecycle of a product such as requirements, calculations, models, test results, marketing material, service documentation, and much more [101]. As a concept, however, PLM is a paradigm that transcends software implementation [78].

The identified gaps in Wavin T&I system development all pertain to the management of information across the product lifecycle and, as such, pertain to product lifecycle management. For this reason, the solution to these gaps will be labeled as a PLM solution, even though the exact definition of PLM may change over time. To conclude, the problem is a lack of product development information management and integration limiting the transition of ICS projects to Wavin T&I as well as their growth. In the following chapters, the development of a proposed PLM methodology is reported, starting with an explorative analysis on related topics, fields, and concepts, followed by an explanation of the development approach. Then, the resulting methodology is described, including its implementation and limits. Subsequently, in the discussion, generalization and application to other organizations of the methodology is covered.

4

Explorative analysis

The explorative analysis was performed to gain insights in existing concepts, research, practices, et cetera, that pertain to the management of information in product development. This chapter is ordered from abstract to applied by covering relevant fields and disciplines first, followed by methodologies, methods, models, tools, and finally, software tools. This analysis provides a top-down perspective on product development, as will be discussed in section 5.2. Items that have been analyzed but were considered less relevant to Wavin T&I product development can be found in Appendix D.

4.1 | Fields and disciplines

There are many fields and disciplines that are related to product development for businesses scaling up new product development departments. Their definitions are often unclear and there is much overlap between them. To provide an overview, a selection of these relevant fields and disciplines, as well as their most significant interrelations, are briefly covered in this chapter. Additionally, a graphical representation is provided in figure 4.1 for reference. In the graphical representation, the fields and disciplines are structured in three levels based on whether the field or discipline is of a more organizational or operational nature.

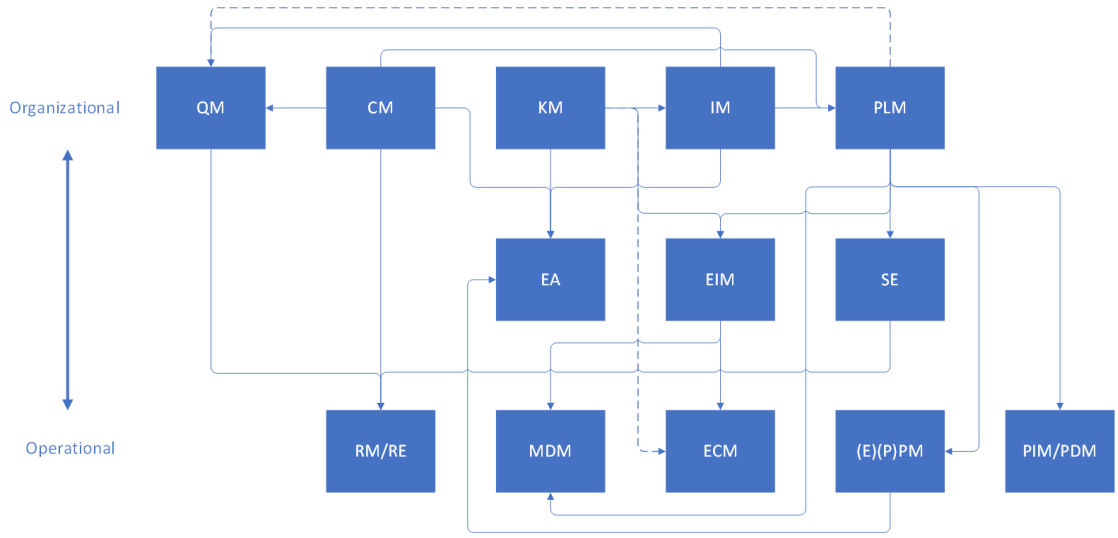


Figure 4.1: Relevant fields and disciplines and their interrelations

4.1.1 | Quality Management (QM)

Quality management is the integrated approach of minimizing product faults. For this reason, by nature, quality management is mostly (production) process oriented. However, the belief that quality can be 'designed in' has resulted in the extension of quality management across the product lifecycle, shifting the focus from product faults to customer satisfaction [129]. This shift emphasizes the importance quality management in the development phase.
Relation to RM/RE: quality management tools and methods elicit many requirements and constraints in product development.

4.1.2 | Change Management (CM)

In a business environment that is constantly evolving and increasingly competitive, change management is a necessity [24]. However, many organizational change management initiatives fail [12]. For companies that develop products or systems, change management can be divided in two main categories: organizational (and strategic) change management, and engineering change management (also referred to as change control). Of these categories, the latter is most relevant in product development.
Relation to QM: The tracking of changes is a component of quality management as well.
Relation to RM/RE: Engineering change management controls the processing of changes to (system) requirements.
Relation to EA: Organizational hierarchy and change management can be captured in an

enterprise architecture.
Relation to PLM: The information that is produced by engineering change management (such as information related to configuration management) is part of PLM.

4.1.3 | (Enterprise) (Portfolio and) Project Managemnent (EPPM)

Project Management (PM) is a well-known discipline that aims to realize the project goals using the available resources under given constraints [105]. Project portfolio management is concerned with the strategic alignment of project management across multiple projects in a project management office [49]. Enterprise project and portfolio management is the top-down approach that governs and coordinates practices and strategies for the PMO and coordinates larger organizational projects [44].
Relation to EA: EA can be complemented by EPPM by applying PPM concepts to the architecture [49].

4.1.4 | Enterprise Information Management (EIM)

Enterprise information management aims to manage unstructured information alongside traditional content, documents and records, such as social interactions, business information and business intelligence [14].
Relation to MDM: MDM is a more business-oriented information management approach [131]
Relation to ECM: ECM is a component of EIM, focusing on the storage, preservation, protection and reusability of information [14].

4.1.5 | Enterprise Content Management (ECM)

ECM, as a component of EIM, is concerned with the integration of structured and unstructured information within a business [73]. The Association for Intelligent Information Management defines the purpose of ECM as capturing, managing, storing, preserving and delivering this information [8]. ECM attempts to combine enterprise resource planning with document management systems, and functions include the provision of collaboration tools as well as content, records, and document management software. Furthermore, opportunities for ECM have been identified with the rise of Web 2.0 (the semantic web) [107]. As the scope of ECM expanded, the notion that ECM solutions would be a combination of multiple (software) tools arose, leading to the development of overarching frameworks [54]. However, with the hype surrounding Web 2.0 dying off [53], largely, so did the term ECM [137].

4.1.6 | Knowledge Management (KM)

Though there is no single definition of knowledge management, it has been defined as follows by Ngai and Chan [92]: “KM refers to the set processes or practice of developing in an organization the ability to create, acquire, capture, store, maintain and disseminate the organization’s knowledge”. The functions in this definition are similar to those of ECM, which results in overlap between the fields. Knowledge is a more abstract concept, which can be extracted from information and content, which is why knowledge management can be regarded as a more abstract field than EIM and ECM.
Relation to EIM/ECM: Since knowledge can be extracted from information, there is much overlap between knowledge and information management. Furthermore, a KM strategy can set requirements for EIM procedures and ECM tools.
Relation to EA: EA is heavily influenced by and can be fully based on business ontologies, which are an important concept in KM.
Relation to IM: The use of a knowledge management system has been correlated with increased knowledge management capacity, which in turn has been shown to support innovative capacity [112].

4.1.7 | Innovation Management (IM)

Gartner defines innovation management as the discipline that drives a repeatable and sustainable innovation process and culture in an organization [48]. According to ISO 56000, innovation management can be performed for several reasons, including alignment of innovation activities and initiatives with the organizational strategy and striking a balance between improving on current performance and exploiting novel opportunities [70].
Relation to EPPM: according to ISO 56000, planning and support are fundamental elements in an innovation management system. These are also fundamental to project (and portfolio) management [70].
Relation to EA: An IM strategy can be implemented in an enterprise architecture [7].
Relation to PLM: The openness of innovation as well as the balance between incremental and radical innovation influences a PLM strategy.

4.1.8 | Requirements Management (RM) / Requirements Engineering (RE)

Requirements management is concerned with the structuring and administration of elicitation, derivation, analysis, coordination, versioning, and tracking of requirements of a product along its lifecycle [57]. Requirements engineering is the systematic approach of

applying requirements management. The term requirements engineering is frequently used in software and systems engineering [99].

4.1.9 | Product lifecycle management

The definition of PLM is covered in section 3.3.
Relation to QM: Quality management has implications for the whole product lifecycle. Furthermore, QM methods can be applied to the product lifecycle [30].
Relation to SE: SE is concerned with systems lifecycle management.
Relation to EPPM: the realization of a PLM strategy and the use of PLM tools and methods is orchestrated by project management.
Relation to PIM/PDM: PIM/PDM is an essential component of PLM.
Relation to EIM: EIM complements PLM [38].
Relation to MDM: Product master data can serve as a single source of truth in PLM.

4.1.10 | Product Information Management (PIM) / Product Data Management (PDM)

Product data management is the process of managing all the data that defines a product, from design to manufacturing to end-user support [130]. PIM is the storage of all information required to sell a product, whereas PDM is its more specification-oriented counterpart. PIM is used in the context of MDM [23]. PDM, on the other hand, is used in the context of PLM [123]. Examples of information PIM manages are suppliers, vendors, and marketing materials. Examples of PDM are specifications, certificates and bills of materials. Together, they encompass the storage of everything there is to know about a product as sold.

4.1.11 | Master Data Management (MDM)

The goal of master data management is to integrate data on products, customers, and suppliers, and exploit its value [131]. It is an operational approach to information management, and it is therefore closely related to EIM. Master data is business-critical information that commonly pertains to partners, customers and transactions of an organization [35]. As such, it is less related to products, the product lifecycle, and the information that comes with it.

4.1.12 | Enterprise Architecture (EA)

Dedić [31] defines enterprise architecture as “a discipline that: i) defines, organizes, standardizes, and documents the whole infrastructure and all-important elements of the respective organization, covering relevant domains such as business, digital, physical, or organizational; and ii) the relations and interactions between elements that belong to those domains, such as processes, functions, applications, events, data, or technologies.”. The Zachman framework, the The Open Group Architecture Framework [127], and the Department of Defense Architecture Framework [133] are common frameworks in enterprise architecture [139].

4.1.13 | Systems Engineering (SE)

The International Council on Systems Engineering [64] defines systems engineering as follows: “Systems Engineering is a transdisciplinary and integrative approach to enable the successful realization, use, and retirement of engineered systems, using systems principles and concepts, and scientific, technological, and management methods.”. It is concerned with all the challenges that arise when developing a system of interrelated components.
Relation to RM/RE: Requirements management of systems is one of the systems engineering processes.

4.2 | Methodologies

As discussed in section 5.1.1, methodologies are a set of methods combined with a set of principles. In this section, relevant and common methodologies in product, system, and software development are covered.

4.2.1 | Design thinking

Design thinking is a common product development methodology that focuses on problem solving through the use of design tools and philosophy. Its important concepts such as empathizing, ideation, collaborative experimentation, and failure acceptance show it to be a mind-centric, intuitive development methodology. It is suitable for organizations with a more collaborative rather than specialized approach, and commitment- over metrics-focused leadership [89].

4.2.2 | Lean thinking

Lean thinking is the overarching term of lean startup, lean product development, lean project management and lean production. Lean thinking, even though frequently applied in manufacturing, is applicable to product development. Hoppman et al. [59] show that lean thinking must be applied to the complete value stream, rather than just the production domain. Intuitively, this seems logical: what is the use of optimizing the manufacturing of a product to make it lean, when the product inherently (meaning by design) is not lean? In product development, lean product development concepts such as workload leveling, cross-project knowledge transfer, supplier integration, and simultaneous engineering can reduce the lead time of projects as well as rework [59].

4.2.3 | Agile

The agile methodology, referred to as Agile Software Development in the context of programming, is a development methodology focused on adaptivity [116]. This adaptivity is mainly achieved through incremental development and verification [34]. The twelve principles of the Agile Manifesto are focused on software development specifically [16], but can be adapted for, for example, embedded systems engineering. Where lean thinking focuses on cutting out unnecessary activities, agile focuses on efficiently and adaptively organizing the activities that remain [34]. The Scaled Agile Framework (SAFe) exists to address manageability issues of agile development. It has its own set of principles, aiming to balance agility with scalability [113].

4.2.4 | TRIZ

The Theory of Inventive Problem Solving (of which the Russian translation shortens to the acronym TRIZ), is a problem-solving methodology consisting of tools and a large set of innovation principles. By formulating a problem in terms of TRIZ concepts, it can be compared to one of the generalized TRIZ problems and their accompanying solutions, which can then be translated to a specific solution for the problem at hand [19]. Central concepts in TRIZ are identification and resolution of contradictions, defining and striving for ideality, and recognizing evolutionary patterns in systems [63].

4.2.5 | Model-based Systems Engineering (MBSE)

Model-based Systems Engineering (MBSE) is the systems engineering approach that applies models to represent system information at different levels of abstraction, without losing control of the details [61]. It focuses on the formalization of models in requirements

elicitation, traceability, design, analysis, and verification of systems [81]. The modeling language most used in MBSE is SysML, which is an extension of UML by the International Council on Systems Engineering [68]. Generally, the use of MBSE has been correlated with improved systems engineering efficiency and quality. However, its learning curve, lack of perceived value and predominance of document-based processes limit widespread adoption [61].

4.2.6 | Academic methodologies

As opposed to the previous methodologies, the following are not endorsed and/or certified by consortia, foundations, or commercial organizations.

4.2.6.1 | Integrated Product Development

Integrated product development is a methodology that places engineering design in its context. At the core stands a model that indicate three tracks that all originate from the same user need: marketing, design and manufacturing. This methodology has largely been superseded by the stage-gate model and lean product development but has indicated a clear need for product development management already in the eighties [4].

4.2.6.2 | Axiomatic design

Axiomatic design is a methodology initially developed to create a scientific basis for the field of design. It entails a design approach based on axioms, which are self-evident truths that cannot be derived or proven true except for the inability to find exceptions. Axiomatic design is divided into four domains: the customer, functional, physical, and process domain. Mapping between these domains translates the what to the how of designs. The two axioms central to the methodology are posited as universal truths in good design [124], serving as starting points for further reasoning.

4.3 | Methods

This section covers relevant product, system, and software development methods. Some may be associated with a previously mentioned methodology. However, many standalone methods exist as well. Furthermore, as discussed in section 5.1.1, the boundaries between methods and methodologies are not always strict. Miscellaneous methods (methods that were deemed less relevant to Wavin T&I product development) can be found in Appendix D.

4.3.1 | Stage-gate

The stage-gate method of product development divides the development process into fixed stages and applies formal structured reviews before a design is allowed to continue. The goal of applying the stage-gate method is to reduce the probability of costly setbacks [139].

4.3.2 | Scrum

Scrum is an agile development method that comprises of predefined roles, meetings and artifacts [116]. Scrum projects are divided into sprints, led by a scrum master, which are small portions of work that can be achieved typically within one to four weeks. This is an example of incremental development improving the agility of development.

4.3.3 | Kanban

Kanban is an agile method that focuses on continuous improvement [116]. Kanban originally consists of emitting a signal indicating readiness for new supplies or new work realizing the pull concept of lean production [96]. In agile product development, Kanban entails creating this pull-based flow to improve development efficiency.

4.3.4 | Continuous integration / continuous deployment

Continuous integration is a popular agile software development practice, aiming to integrate software parts frequently, throughout all phases of the development cycle [94]. Continuous integration is frequently mentioned and applied in conjunction with continuous deployment, which is the process of streamlining the deployment of changes to software up to the point where changes can be automatically deployed.

4.3.5 | Quality function deployment

Quality function deployment (QFD) is a method that entails applying the house of quality (HoQ) tool over multiple phases. In the first phase, the customer needs are cross-referenced with the available technical measures. In the second phase, the technical measures are matched with parts characteristics. These are matched with process operations in the third phase, and finally with production requirements in the fourth phase [28]. The second, third, and fourth QFD phase are frequently omitted in practice: often, only a single HoQ is used to couple customer needs to product features.

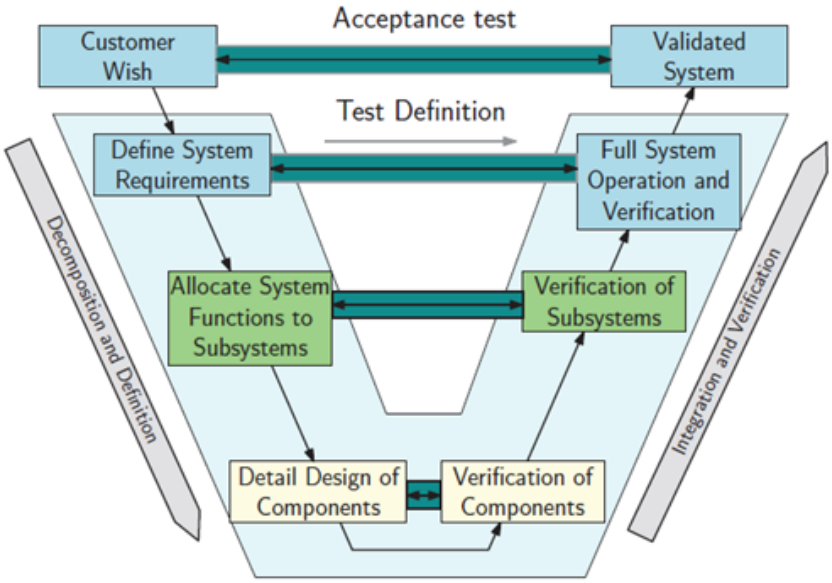


Figure 4.2: Extended V-Model [19]

4.3.6 | Morphological analysis

Morphological analysis is a procedural method used to explore combinations of working principles [139]. It uses the morphological chart, also called morphological matrix or Zwicky box, in which candidate solutions are listed against non-quantifiable parameters [108].

4.4 | Models

This section covers models that are frequently used in conjunction with or form the cornerstone of development methodologies and methods. Miscellaneous models (models that were deemed less relevant to Wavin T&I product development) can be found in Appendix D.

4.4.1 | V-Model

The V-Model (also called Vee model) is a common process model of systems engineering, highlighting the process of architectural decomposition and integration. Verification and validation at each (sub)system level ensures that integrity and consistency is maintained when traversing from decomposition and definition to integration and realization [19, 139]. The V-Model, extended by Bonnema et al. [19] is shown in figure 4.2.

4.4.2 | Kano model

The Kano model plots customer satisfaction against degree of implementation of a product or service feature [74]. This allows for the classification of features into categories such as basic, performance, and excitement features. Basic features result in a negative satisfaction when not implemented, but to not yield increased customer satisfaction if they are. Excitement features do not negatively impact satisfaction if not implemented but do yield increased satisfaction if they are. Performance features can both positively and negatively influence customer satisfaction along varying degrees of implementation.

4.4.3 | Double diamond model

The double diamond model is a design process model that distinguishes between four phases: discover, define, develop and deliver. The first and third are diverging processes, whereas the second and fourth are converging processes, resulting in a double diamond shape (shown in figure 4.3). This model illustrates the modes of thinking of designers along the design process [32].

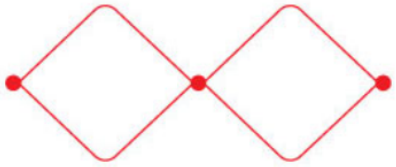


Figure 4.3: Double diamond process model [32]

4.4.4 | SECI knowledge transfer model

The SECI knowledge transfer model structures the various types of knowledge and their conversion. The model distinguishes between tacit and explicit knowledge, where tacit knowledge resides in a person's mind, and explicit knowledge is stored in an artifact. The exchange of tacit knowledge is called socialization, and the exchange of explicit knowledge is called combination. The conversion of explicit to tacit knowledge is called internalization, and, finally, the conversion of tacit to explicit knowledge is called externalization. Socialization, externalization, combination, and internalization together form the SECI acronym [95].

4.4.5 | Academic procedural design models

There are numerous well-known models of the design process, such as those of Pahl and Beitz, French, Pugh, and Ullman [139]. These typically present the design process as sequential, (partially) parallel, or iterative. Design research and the process models that it results in have formed the basis of many commercial design process models, methods, and methodologies over the years.

4.5 | Tools

This section covers common tools that can be applied in previously mentioned methods or methodologies, as well as in product, system, and software development in general. Miscellaneous tools (tools that were deemed less relevant to Wavin T&I product development) can be found in Appendix D.

4.5.1 | Failure mode and effect analysis (FMEA)

The Failure Mode and Effects Analysis (FMEA) is a tool that systematically quantifies potential failures of a design or a process. Failure modes are the 'ways' in which the design or process can fail. The occurrence, severity, and detectability are assigned scores on a scale and subsequently multiplied, yielding a risk priority number [19]. FMEA is commonly applied in analysis of the reliability of a system [34].

4.5.2 | Obeya room

The Obeya room is a lean thinking tool in which a 'war room', so to speak, is employed to provide frequent, quick, and effective status updates to chief engineers, project managers, and senior management [59]. The Obeya room is an example of visual management, which facilitates the management of sequential project steps and occurring bottlenecks [139].

4.5.3 | PESTLE framework

The PESTLE framework (also arranged as PESTEL) is a framework comprising of the following set of contextual factors: political, economic, social, technological, legal, and environmental [122]. The use of this framework enables the structured consideration of all the contextual influences that act on a business, department, or system.

4.5.4 | Lessons learned log

The lessons learned log is a knowledge management tool that facilitates reuse and dissemination of the knowledge gained during a project. It is a log that documents what went right and what went wrong, as well as the implications for future projects and is made in the termination phase of a project [100].

4.5.5 | Responsibility matrix

The responsibility matrix or RACI chart is a tool used for human resource management in projects [100]. In a responsibility matrix, one of the following degrees of involvement is assigned to project members or roles: responsible, accountable, consult, inform, from which the acronym RACI originates. Every activity should have at least one responsible person, whereas accountable roles and roles that should be consulted or informed are optional [105]. The responsibility matrix provides a clear and unambiguous overview of responsibility and involvement of project activities.

4.5.6 | SWOT analysis

The SWOT analysis is a tool in which strengths, weaknesses, opportunities, and threats of a project or organization are identified [122]. SWOT analyses can be used from a marketing perspective, but also from a project management perspective. In project management, a SWOT analysis is a component of risk management [100].

4.5.7 | Risk register

The risk register is a document consisting of a list of identified risk, their potential owners, and their potential responses [105]. Where a decision tree analysis is a useful tool in the planning phase of a project, a risk register is useful for monitoring risks [19].

4.5.8 | Power-interest matrix

Originally developed by Mendelow [84], the power-interest matrix is a tool that prioritizes stakeholders in a project. The axes of the matrix are, as the name suggests, power and interest. Power indicates the influence a stakeholder has on the project, whereas interest indicates the influence the project has on the stakeholder. The power-interest matrix is used in the stakeholder analysis of chapter 2.5.

4.5.9 | Schedule management tools

Common examples of schedule tools are Gantt charts, schedule crashing, and PERT [100]. These aim to visualize and structure the activities of a project to achieve the project goals within time constraints. Additionally, the project evaluation and review technique (PERT) facilitates in contingency planning through alternative schedules when employing the critical path method [139].

4.6 | Software

Aside from the software that is used to perform product development itself (such as (E)CAD, IDEs, and simulation software), there are many programs that support the knowledge and information management of the development process. In this section, classes of relevant software in product development management are identified and common offerings are discussed.

4.6.1 | Project management

There are many programs that support project management for all business sizes. The most popular project management tools work out of the box, but provide limited flexibility such as Monday.com, Atlassian Jira, and Asana. Solutions aimed at larger organizations such as Planview PPM Pro and Oracle Primavera are more complex in their rollout but provide better scalability and flexibility. The core features of project management software are work assignment, deadline and schedule management, and dependency management of work. Azure DevOps is a one-stop-shop product for many software development related practices such as version control and testing management by Microsoft that also provides these project management features.

4.6.2 | Requirements management

Requirements management programs are commonly used in the medical device industry. Typically, they focus on formalizing the definition of a system. Examples are IBM Rational DOORS, Modern Requirements 4 DevOps, Matrix ALM, Jama, and Visure. In the requirements management software domain, the tradeoff must be made between user friendliness, integration, and ubiquity. For example, IBM Rational DOORS is a complex requirements management program with an outdated interface, but with thorough documentation and support. On the other hand, Modern Requirements 4 DevOps provides a novel user interface and integrates fully with Azure DevOps.

4.6.3 | Systems modelling

Systems modelling software is the cornerstone of model based systems engineering. Most system modelling applications support the SysML language. Examples of modelling software with a focus on MBSE are Modelio, Cradle, Cameo Systems Modeler, IBM Rhapsody, Catia MagicDraw, Vitech Genesys, Capella, and Innoslate. With the exception of Innoslate, these tools suffer from outdated interfaces and limited online collaboration. Capella promotes the application of their Arcadia systems engineering method over SysML.

Features supported by some of these solutions are system state simulation and the modification of the underlying metamodel, which is the definition of objects in models and their possible relations. Common application of systems modelling tools is in highly complex, safety-critical systems such as in the aerospace industry.

4.6.4 | Enterprise architecture

Enterprise architecture software is a type of specialized modelling software for defining the components and processes of an operation. The most common EA software, Sparx Enterprise Architect, supports languages such as UML, SysML, and OWL, as well as various enterprise architecture frameworks like the Zachman, DoDAF, and TOGAF frameworks. Archi is a free, open-source alternative that enables architecting in their proprietary Archimate standard. Finally, Planview Enterprise One is a project and product portfolio management suite that supports basic planning of enterprise architectures. Due to the specialized nature of EA and the possibility to perform EA in other modelling software, there are not many dedicated EA software solutions that gain widespread popularity besides Sparx Enterprise Architecture and iServer Suite.

4.6.5 | Product lifecycle management

Product lifecycle management software suites are often large scale solutions for large scale manufacturers of complex products. Examples of PLM suites are PTC Arena and Windchill, Siemens Teamcenter, Oracle Agile PLM, Aras Innovator, SAP PLM, and IBM Engineering Lifecycle Management. PLM software suites are commonly built on core functions of requirements management, change management, systems modelling, and product data management. They play an integrating role in the management of all product and process information, tailored to the organization scale and product complexity, often resulting in lengthy procurement and rollout procedures.

4.6.6 | Commonalities

The common element in almost all these programs is that they are linked database management programs, with varying flexibility and interface user-friendliness. Since most of these tools store their data in XML format and provide REST APIs, exchange with other software is often possible, albeit at varying levels of ease. Furthermore, licenses of these tools are generally expensive, especially at higher subscription tiers that provide more support and flexibility. Many vendors of these software categories also provide product variants target at varying organization sizes, as well as multiple products covering dif-

ferent areas within product development information management. Even though, as said before, exchange of information between programs is possible, software vendors attempt to sell a software ecosystem, occasionally hindering interoperability.

5

Approach

5.1 | Methodology

In this thesis, the terms methodology, method, tool, model, and framework are used extensively. Even though the definitions frequently overlap and occasionally can be used interchangeably, working definitions for clarity are presented in this section. Subsequently, the structure of the methodology of this thesis is explained.

5.1.1 | Definitions

Model
In the abstract sense, a model is a description (or depiction) of a system or process used for calculations or predictions[1].

Method
A method is another word for a procedure[2]. Methods are often a prescription of actions, often in a certain sequence. Many product development methods are the prescribed application of a product development model.

Methodology
Literally, a methodology can be translated as a study of methods. Practically this can be defined as a system of methods, rules and postulates[3].

[1] https://dictionary.cambridge.org/dictionary/english/model
[2] https://www.merriam-webster.com/dictionary/methods
[3] https://www.merriam-webster.com/dictionary/methodology

Tool
In the broadest sense, a tool is something (abstract or physical) that helps in performing an activity[4]. Tools can be employed in the application of methods and methodologies.

Technique
A technique is defined by Merriam-Webster as a body of technical methods[5]. In that sense, the term resides between method and methodology. However, the term technique can also be used to indicate a way of achieving a goal. This definition is useful to denote the application of a method using a tool, placing it below the term method in a hierarchical sense.

A visualization of the hierarchy of these terms is provided in figure 5.1.

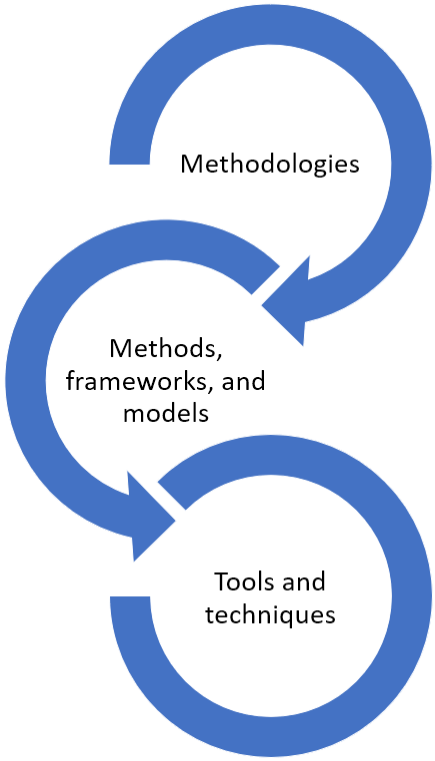


Figure 5.1: Hierarchy of terms, from methodology to tools

5.1.2 | Structure

Applying the stated definitions, the methodology of this thesis is a set of methods bound by principles, addressing the identified gaps using management and software tools. This structure is illustrated in figure 5.2. The result is a selection of curated and developed methods and principles prescribed to Wavin T&I to improve the development management practices for the ICS product line.

5.2 | Application and implementation

The development, application, and implementation of the methodology is realized through a bidirectional approach: existing methodologies, methods, et cetera are applied to Wavin product development from a theoretical top-down perspective, and practical considerations are abstracted to tailor or develop methods from a bottom-up perspective. This

[4]https://dictionary.cambridge.org/dictionary/english/tool
[5]https://www.merriam-webster.com/dictionary/technique

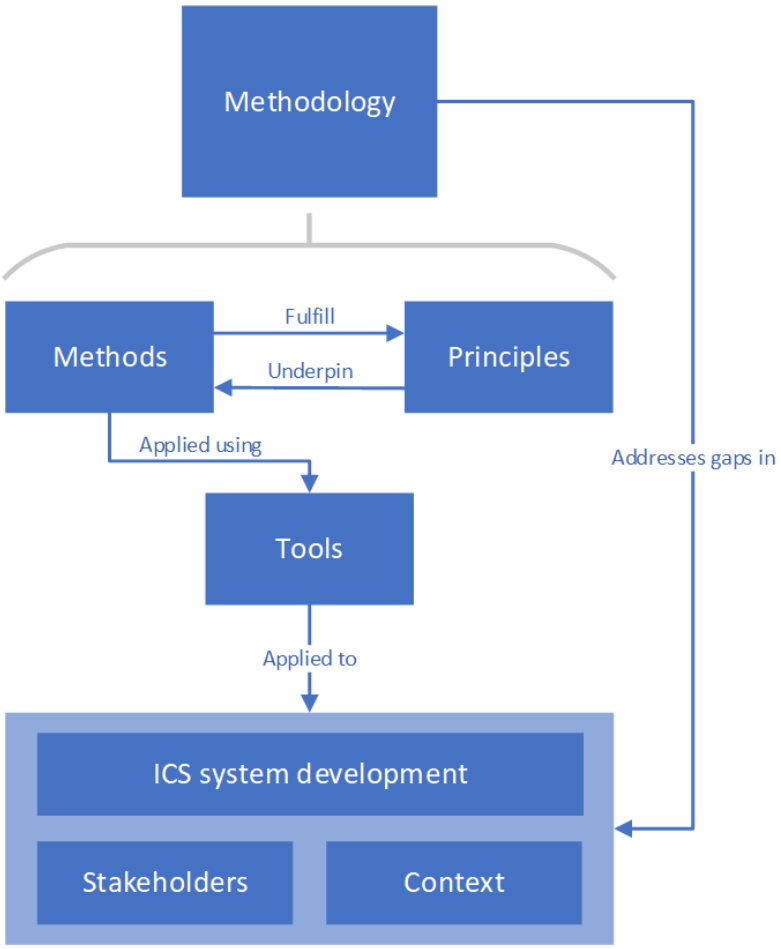


Figure 5.2: Methodology structure

approach not only ensures relevance by being based on current development issues, but also facilitates stepwise implementation, as opposed to a large set of idealized instructions and tools being forcibly inserted into an operational development process. Implementation during development of the methodology, furthermore, provides direct feedback on the support and effectiveness.

6

Methodology

6.1 | Structure

This methodology addresses the gaps in ICS system development using methods bound by overarching principles. As it is a PLM methodology, the methods, principles and tools pertain to the information gathered, produced, and applied across the product lifecycle. This structure is illustrated in figure 5.2 of section 5.1.2.

The methodology is divided into the following components:

- Knowledge management
- Requirements management
- Verification and validation
- Document and configuration management
- Change management
- Prioritization
- Risk management and contextualization

For each of the components, relevant methods, theories and frameworks are assessed, as well as their application to Sentio product development.

6.2 | Scope

The PLM methodology is aimed at system development at Wavin T&I, to address the gaps that limit the transition of development on request to development ownership. However,

aspects of product development rarely frequently transcend the core product engineering team, and the Sentio project is characterized by extensive collaboration with development partners. In this section, the relations between the ICS business unit, PLM, external development partners, and the project management office (PMO) will be outlined.

6.2.1 | Development and testing

Development of ICS products is coordinated by Wavin T&I. However, the execution of the development effort (writing code, designing PCBs) will remain mostly handled by development partners such as Jablotron. The scheduling, monitoring, and support of this process is the responsibility of the project manager, whereas the management of its related information is within the scope of PLM. The same applies to verification and validation. If or when development is delegated to more partners, the complexity of the project manager's role increases, as well as the potential benefit of PLM practices.

6.2.2 | Risk management

Risk management, and the monitoring of contextual factors in general, lies at the intersection of the PMO, BU, and PLM. Development risks and contextual factors are inherent to the project and should therefore be monitored by all those involved in the project. For example, a risk may be identified by the BU, after which a risk response strategy can be developed. The PMO is then responsible for the implementation and monitoring of this strategy. PLM can facilitate this entire process.

6.2.3 | Change management

PLM supports the management of information of changes across the product lifecycle. However, the change management process must be fostered by the BU. Granting T&I the authority to manage engineering changes is an organizational change in itself.

6.2.4 | Products

The scope of the methodology is on the development of systems in the Sentio product line, meaning products that comprise of subcomponents, each with hardware, firmware, and software. In short, the scope of the PLM methodology is system development management at Wavin T&I.

Table 6.1: Components addressing gaps

Component	Gaps
Knowledge management	3, 4, and 7
Requirements management	1 and 2
Verification and validation	5
Document and configuration management	1 and 8
Change management	2, 3, 7, and 8
Prioritization	1, 3, and 8
Contextualization and risk management	6

6.3 | Components

The components of the methodology contain methods selected and developed to address the system development gaps. For every component, its application to Wavin product development is discussed, indicating how it covers its associated gaps. These components cover the accompanying metamodel and are underpinned by the seven central principles covered in subsequent subsections. In table 6.1, the gaps each component addresses are shown.

6.3.1 | Knowledge management

6.3.1.1 | Knowledge management in product development

Identification and assessment In product development, artifacts of knowledge most importantly reside in specifications, requirements, decisions, investigations, documentation, lessons learned logs, assumptions, among many more. Naturally, the knowledge stored in artifacts does not correspond exactly to the knowledge in the mind of any individual. Still, identifying which knowledge is captured in which artifact, and assessing their quality will improve product development performance; the use of KM (software) tools and techniques such as PLM and PDM systems, visual management, and lessons learned documents has been positively correlated with KM performance when measuring ease of entering, retrieving, and reusing information [13].

Organization The organization of knowledge fully defines its reusability at the cost of quality and capturing effort. Product development knowledge can be organized in several concepts. The first and most obvious is textually, which requires little knowledge as a technique, but often times does not adequately capture knowledge. The second is topic and concept maps, which are overviews of relevant concepts with connotated lines indicating their relationships. Conventions of topic maps are standardized in ISO 13250

[66]. They are of a descriptive nature, with an emphasis of findability. The third concept is ontology. Ontology is the philosophy of being: ontologies define conceptual knowledge explicitly [22]. Ontologies are internally consistent vocabularies [29], meaning that automated semantic reasoning can be applied to them [141]. Ontologies of and for project management and product development have been made [46, 3, 55], but their practical application is negligible. Metamodels are the fourth knowledge organization concept. According to Bakirtzis [11], metamodels define the types of information and their relations in a conceptual system or process. They can be regarded as an abstraction of models. Metamodels are more prescriptive in nature, meaning that the truth can be constructed from the model, whereas ontologies are descriptive, meaning that the ontology describes and classifies the truth [142]. In other words, ontologies are more focused on representing the real-world concepts truthfully and consistently (semantically), and metamodels are focused on application-oriented (syntactical) representations of the real-world concepts [111]. The fifth, more practical concept in product development knowledge organization is a (design) knowledge catalogue [98]. Design catalogues store previous technical solutions and design practices to facilitate their reuse [90]. An example of their reuse is in QFDs or morphological analyses in subsequent projects.

Dissipation and representation The sharing of knowledge is regarded as the most significant issue in knowledge management [47]. It has been positively correlated with increased customer value and reduced time to market [58]. The exchange of knowledge is commonly discussed in terms of the SECI knowledge transfer model [91], which is covered in subsection 4.4.4. Notably, internalization (the transfer of explicit to tacit knowledge) and socialization (the transfer of tacit knowledge) have been shown to play a significant role in product development [91]. Wu et al. [138] have developed a method for knowledge integration and sharing that, among others, shows relevant information based on similarity using an underlying product development ontology. This method, however, remains only highly experimental, and is limited to the knowledge of engineers specifically [138]. Nevertheless, the premise of providing relevant product (development) information based on semantical similarity is promising for the future. More applicably, visualization has been shown to improve knowledge sharing [25]. Still, the process of visualizing knowledge for communication is that of externalization and does not fulfill the significant role of internalization and socialization mentioned earlier. In general, it can be concluded that both formal and informal exchange of knowledge occurs in product development.

6.3.1.2 | Application

Structuring system developments around a metamodel is suited for product development at Wavin, as it strikes a balance between an abstract and pragmatic organization of knowledge. The fact that a metamodel, opposed to an ontology, can be easily visualized also adds the benefit of communication; an overview of the concepts in product development can be gained at a glance. The contents of this metamodel are covered in section 6.4. Additionally, investigations performed at Wavin T&I should be reported to enable the communication and reuse of the acquired knowledge. As the development endeavors of Wavin T&I scale up, so too should their knowledge management efforts. In the future, for example, the implementation of design catalogues in the form of best practices and standardized technical solutions could provide rapid benefits in knowledge management performance.

6.3.2 | Requirements management

6.3.2.1 | Basics

In system development, requirements are often a system as complex as the product itself, quickly exceeding what a single person can oversee [57]. Requirements management then becomes a necessity to develop a product that adequately fulfills the needs of the end users. Requirements engineering is the practice that encompasses the complete lifecycle of requirements [99], as illustrated in figure 6.1.

Requirements are commonly divided in to functional and non-functional. Functional requirements specify the functions and actions a system must perform. Non-functional requirements, although debated, are generally accepted to concern '-ilities' such as reliability, usability, and constraints such as volume and speed [51]. Requirements elicitation is the process of gathering all requirements from a technical, user, and contextual perspective [99].

Storing requirements Reviewing system requirements is essential to develop a system that the commissioning party (senior management or a client) envisioned [86]. Requirements are generally stored textually in lists, commonly referred to as requirements specifications. Basing requirements on models rather than on text and semantics can aid in creating a shared understanding [86]. Still, consensus must be reached on the models for requirements engineering. Ensuring that a set of requirements is truly complete is impossible, but measurable consistency has been shown to improve requirements completeness [77]. From this we can derive the necessity of traceability in requirements. Graph

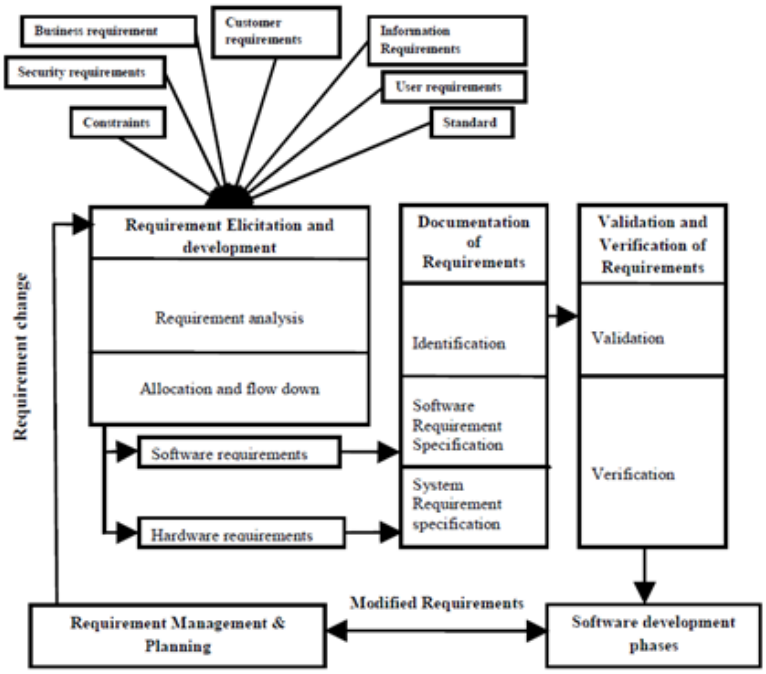


Figure 6.1: Requirements engineering process model [99]

based databases can store requirements and their interrelations effectively. Visualization, traceability and test case coverage have been shown to be enhanced by this method [80, 6]. However, this technology is in its early stages and not available in commercial requirements management tools. Nevertheless, it further emphasizes the importance of the interrelations of requirements.

Requirements, agile, and lean When adhering to the agile methodology, requirements are captured in the form of user stories. These interrelate to one another, introducing difficulty in traceability when compared with the traditional waterfall methodology [26]. Still, user stories support requirements development from a user perspective rather than a technical perspective. In simultaneous engineering, concurrently with defining requirements, their verification and validation should be developed. This is also part of lean product development, where the most important aim is to reduce waste [59], including wasted effort specifying verification and validation plans retrospectively.

6.3.2.2 | Application

Requirements traceability should be an essential component in engineering practices at Wavin T&I, as it improves project performance [83]. However, a tradeoff must be made

between required effort to fully implement end-to-end traceability and its benefits, since it is not required like with standards in the automotive industry such as ISO 26262 [69] and IEC 61508 [62]. However, traceability is endorsed for quality assurance in ISO 10007 [65] and ISO 9001 [71]. In Wavin projects, functional requirements should originate from change requests, which are based on user stories, to ensure end user benefit for every implemented function. Non-functional requirements should be elicited from contextual factors and risks and can be traced as such. Matrix ALM is a software tool for requirements management that Wavin T&I already has access to. It supports the implementation of custom schemas, meaning that the nature of requirements, for example non-functional and functional, and their traces to other artifacts can be fully customized. Furthermore, it facilitates the workflow for reviewing requirements and exporting them for communication. A previously conducted Calefa quality project has resulted in the initiation of a Quality Management System (QMS) for Calefa. Lessons learned and infrastructure from this project can be applied to develop a QMS for Sentio. For example, as Wavin aims to take (more) ownership of technical Sentio development, more regulatory requirements will be bestowed upon them. Preparing for (self-)certification is therefore necessary, for which a QMS is essential.

6.3.3 | Verification and validation strategy

Verification and Validation (V&V) is the process of verifying whether the designed product meets the requirements, and whether the products and the requirements meet the envisioned user needs [19]. While V&V is a big component of quality assurance, the two are not the same: V&V is one of the components of achieving the quality defined in the quality management strategy of a company. In this subsection, verification and validation methods and considerations relevant to Wavin as they increase their systems and software development endeavors are discussed.

6.3.3.1 | Activities

V&V in software engineering In software development, the testing activities can be divided into two main categories: black box testing and white box testing [93, 75, 43]. Black box testing, also called functional testing, is applied to finalized software products, and verifies the code from a user perspective [93]. The tester will not have access to the source code, resulting in relatively easy test prescriptions. An important black box testing technique is state transition testing [75]. White box testing, also called structural testing, is when the test cases are developed based on the source code and verifies whether the code functions as intended [93]. White box testing can uncover implemen-

tation errors on a unit, component, integration, and system level [75]. Important white box testing techniques are integration testing and regression testing. Integration testing verifies the functioning of the individual components as they are placed in context, and regression testing verifies the functioning of the rest of the system as changes are made to one component [93]. The trivial combination of white and black box testing is called gray box testing. Model-based software testing can be seen as a form of gray box testing [75]. Software test automation is a component of software verification and validation that can yield significant benefits in the form of time saving, test coverage, and effort saving. However, common limitations in the implementation of automated software testing are high initial cost of designing test cases, high test case maintenance, and purchase of test automation tool(s) [106].

V&V in systems engineering Verification and validation activities of systems can be divided into five classes [118]:

1. Verification by similarity: the system is verified by comparing it to already verified systems.
2. Verification by analysis: the system is verified by simulations and/or analytical data.
3. Verification by testing: the system is verified by executing predefined test plans.
4. Verification by demonstration: the system is verified by operation in a real world scenario.
5. Verification by examination: the system is verified by direct measurements.

The decision between V&V activities can be made based on quality, cost, and risk [118]. One of the main benefits of MBSE is the possibility to execute V&V activities early in development. For example, models allow for simulation and integration testing (of components and their interfaces), while it is still possible to implement more significant changes [87]. In other words, at various levels (unit, component, integration, and system), across the development process, different classes of V&V activities should be considered [82]. Additionally, responsibility of V&V activities should correspond to development responsibility in general, meaning that those who are responsible for development of a component are also responsible for its verification. This requires trust in and extensive collaboration with development partners and may take time to achieve.

6.3.3.2 | Application

As the significant majority of software defects originate from design [43], it is essential to incorporate V&V as early as possible in development. Wavin has the opportunity to incorporate V&V practices in the development of their systems development process itself. As development responsibilities are shifting from development partners to Wavin, Wavin's V&V efforts should grow. To realize this transition, a clear distinction must be made between component (subsystem) definitions at development partners and the overall system definition at Wavin. Generally speaking, Wavin should be concerned with black box testing, *validating* that the right system has been developed and white box testing for integration and regression tests. Subsystem teams should develop and execute their own testing strategies, *verifying* that their components are developed correctly. Test cases for system-level requirements should be developed incrementally, meaning that new and updated requirements should receive test cases first, rather than retrospectively developing test cases for requirements with which no quality issues are experienced. Furthermore, resolved issues (bugs) should result in test cases as well, ensuring the detection of the issue in the future. Over time, this will lead to a continuous increase in test case coverage. Test cases should be designed for reuse in regression testing, to increase verification and validation efficiency, and prioritized by assessing their execution costs, time, and effort [33]. The added benefit of designing reusable test cases is the suitability for software test automation. Test plan development and manual test execution can already be performed in the Matrix ALM requirements management tool of Wavin. This allows for the development of reusable test plans immediately, facilitating the scaling up of the development efforts at Wavin T&I. White box testing and test automation, on the other hand, must still be implemented from scratch if desired after thorough cost-benefit analysis.

6.3.4 | Document and configuration management

6.3.4.1 | Basics

Data and file types in product development In research and development, many types of office files document the process, decisions and results. Typically, these are in the form of reports, presentations, and meeting minutes. More formal static documents are lists of requirements, (textual) design specifications, and risk assessments. These documents are stored in a cloud environment for collaborative capabilities and the archival of previous versions. The development of mechanical designs results in CAD files, which are currently stored at Jablotron in the case of the Sentio product line. These could be stored in a PDM service of Wavin such as SolidWorks PDM for availability and traceability

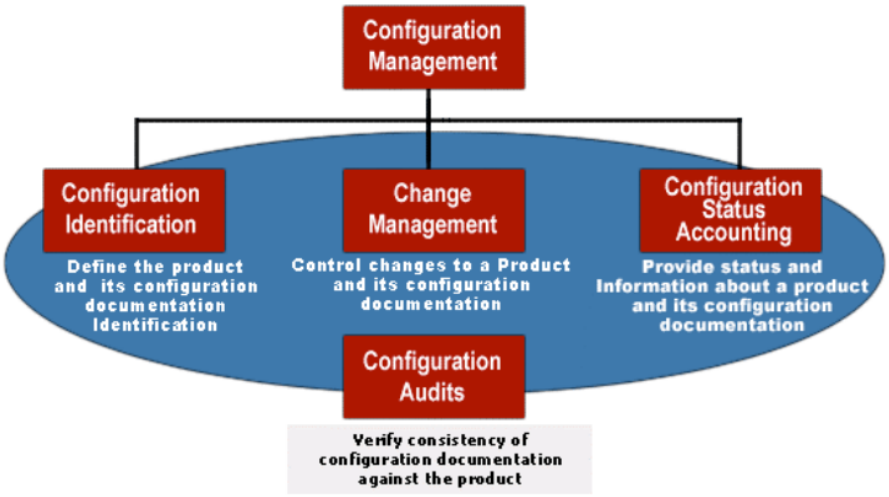


Figure 6.2: Conventional configuration management [135]

purposes, as well as automatic version management. Electronics are developed in ECAD software, such as Autodesk's Fusion 360 or Altium's Designer. Both of these offer their own PDM solution for hardware. Even though web PDM applications have been shown to improve supply chain linking in the past [130], many MCAD/ECAD suppliers wish to provide a complete CAD+PDM ecosystem that only works best when complete, leading to files being scattered across systems, such as, for example, CAD files in SolidWorks PDM, specifications in Microsoft SharePoint, and PCB designs in the Fusion 360 cloud. Software and firmware are developed and stored in a completely different manner: code is written in various languages, using many libraries, compilers, and drivers. For the Sentio product line, source code is stored in a version controlled repository, to which Wavin does not yet have access, due to the use of proprietary specifications of Jablotron. However, with the intention of further taking responsibility of Sentio development, Wavin should aim to separate the source code from the intellectual property of development partners. To transition from Wavin-requested development to Wavin-directed development, distinctions between the system and its subsystems must be made. This will facilitate the discussion on the boundaries of development responsibility and intellectual property with development partners.

Configuration management ISO 10007 defines configuration management as an activity that applies technical and administrative direction over the lifecycle of products and services, and their configurations [65]. The main configuration management activities are configuration identification, change management and configuration status accounting [65, 135], as illustrated in figure 6.2.

Configuration identification

Configuration identification is a continuous process that is supported by version control systems. However, it is essential to identify and clearly define the configuration items [135] as early as possible [65]. These relate subsystem components (mechanical, hardware, firmware, and software) and their versions to product and service versions, along with their interfaces.

A version control system is a system that stores historical versions of project files, usually source code, as well as specific information per version such as changes made, identity of the person that made the change, and comments [109]. Different versions can be worked on in parallel, commonly referred to as branches. In version control systems, the distinction is made between centralized, where the files are stored in a single location, and decentralized, where every user can have their own version of the files [144]. Git is a popular decentralized version control system, commonly accompanied by a well-documented workflow of managing branches, which can be merged into a master branch using so-called pull requests [36, 10]. Recently, trunk-based development has gained popularity over the still frequently used Git workflow [10]. Trunk-based development focuses more on working on a central team-shared branch named the trunk, off of which small branches are split for specific releases. This enables continuous integration and continuous development [56].

Versions of configuration items should be assigned unique numbers. A common numbering system is Semantic Versioning 2.0.0, which employs a “Major.minor.patch” numbering scheme. The increment of the major number indicates changes that are not compatible with previous major versions, whereas minor and patch version increments should be backwards compatible [103].

Version control of general documents and hardware files is often performed ad hoc, resulting in file names with almost meaningless endings such as ‘_final_v2_definitive’, which often significantly increase the difficulty to find the latest relevant version. Cloud storage and PDM systems can automate versioning by automatically saving previous versions, but they do not always suppress the file naming tendencies of engineers, and therefore require agreements on file storage such as naming conventions. Furthermore, Semantic Versioning could be applied on these file types as well.

Change management

Change management in the context of configuration management refers to engineering change management. It is covered in subsection 6.3.5.

Configuration status accounting

Configuration status accounting is the process of managing documentation of configura-

tion items and tracing the documentation (e.g., design output such as specifications) to the configuration information (e.g., requirements baselines). Maintaining this traceability yields two of the largest benefits of configuration management: quality management and regulatory compliance.

Quality management and regulation Traceability is an essential component of quality management systems, as emphasized in ISO 9001 [71]. Regulation both in Europe and in North America even requires traceability for certain groups of products such as medical devices, standardized in ISO 13485 [67] and 21 CFR 820 [1], respectively. Additionally, in Europe, a so-called technical file is required for more types of products, including those in the Sentio product line, to obtain a CE marking. A CE marking is required to sell products on the European market [42]. 21 CFR 820 specifies the use of and requirements for the following three files [1]:

- Design history file (DHF): a file that indicates that the final product design corresponds to the approved requirements for every version of every product (820.30(j)).
- Device master record (DMR): a file that contains device, production process, quality assurance, packaging, installation, and maintenance specifications of the product as-sold (820.181(a-e))
- Device history record (DHR): a file that shows for each produced batch, lot, or unit of a product that it is manufactured in accordance with the DMR and DHF (820.184)

The ISO 13485 standard on medical devices combines the DHF, DMR, and DHR into what is called a medical device file, serving roughly the same purpose [67]. The technical file can be regarded as the European counterpart of the DMR, with varying requirements depending on the directives that apply to the product type. CE marking for some directives requires more detailed documentation, as well as inspection from a Notified Body. Conformity assessment from a Notified Body is not required for the Sentio product line [42]. For products in the Sentio line, the technical file must contain the following:

- Specifications, drawings and explanations of the product
- Specification of standards the product adheres to
- Test reports
- Declaration of conformity

Based on the following directives:

- Low voltage directive [40]
- Radio equipment directive [41]
- Electromagnetic compatibility directive [39]

Harmonized standards can be applied to facilitate this process, after which a declaration of conformity can be signed and submitted [42]. In practice, these files become very large and may contain references to other documents to limit their individual sizes.

6.3.4.2 | Application

Engineers at T&I are experiencing difficulties in finding relevant files and versions. Furthermore, T&I projects are all stored in one SharePoint library with no explicit version control and a predefined folder structure that is not tailored to the project at hand. This folder structure originates from Wavin's stage gate development process but often, files do not exactly fit in these categories, leading to organic folder structures partially intertwined with and partially separated from the predefined folder structure. Furthermore, specifications of products currently on the market are difficult to find, if available at all. This unstructured growth of project information comes natural with the growth of the ICS operation. However, if Wavin is to take on more development responsibility, more configuration management structure must be applied for regulatory purposes.

This can be achieved by basing file storage on the distinction between project-bound and project-transcending files. For project-bound files, the notion that definitive specifications are formal, and should serve as a single source of truth should be incorporated. This should result in a release library in which all design output of products currently on the market can be found. They should be traceable to their requirements as much as possible. This release library must be agreed upon with stakeholders to be the leading definition of the product, and as such, few project contributors should have write access. The boundary between system specifications at Wavin and subsystem specification at development partners such as Jablotron must be defined as clearly as possible, so that at all times, the responsibilities and locations of (sub)system definitions are known. Project-transcending files should be stored independently of projects they may have originated from, to facilitate reuse of previously gathered knowledge.

6.3.5 | Change request management

6.3.5.1 | Basics

Change requests A change request is an artifact that pertains to fixing of an issue, improvement of performance, or the implementation of a feature. The change request is regarded as the primary unit of work in many software development projects [27]. Furthermore, change request management is regarded as a subdomain of configuration management [114, 126]. In the ICS operation, change requests come from many directions. Installers and end users contact Wavin (including various OCs) with issues and questions. Additionally, installers occasionally contact ICS at T&I directly for technical support. Sometimes, they express needs or wishes in these interactions. Besides these direct feature requests from the customers, change requests can come from developers and development partners as well. Their expected benefits can be direct as well as indirect; indirect meaning benefits to developers eventually tracing to benefits for the customer. The change request process is a continuous one, even though it sometimes pertains to a specific system or component version.

Change control board The Change Control Board (CCB), also referred to as the configuration control board, is a forum that organizes and processes change requests [21, 126]. The change control board should consist of members with insight of the development process, knowledge of system architecture, and decision authority. It should have meetings at regular interval to process the backlog of change requests [21]. The frequency of these meetings should be based on the following two factors. On the one hand processing change request in bulk is more efficient [88], and CCB members may not have time to meet as frequently as multiple times per week. On the other hand, letting the change request backlog grow too large makes the meetings cumbersome and leads to the risk of important potential changes going unnoticed or ignored, as decision fatigue quickly arises when too many change requests have to be decided upon in a single sitting. Employing a change control board will improve the scalability of ICS product development while maintaining a fair change request process in which every potentially valuable idea is heard.

6.3.5.2 | Application

Cavalcanti et al. [27] state that processing change requests is a time consuming effort. They also provide the basis of a change request workflow, emphasizing that discussion prior to acceptance and implementation of changes is essential. Xu et al. [140] state that change based configuration management is agile and minimizes required configuration management efforts. The required of processing change requests can be limited by

streamlining the discussion process using a structured CCB meeting as well as a prioritization strategy (covered in section 6.3.6). The combination of these items results in the following general change request lifecycle:

1. Change request is formatted
   Any change request may be logged in proper format, including the following fields:
   - Origin: where did the change request arise?
   - Direct beneficiary: who will profit directly from implementing the change?
   - Customer value: what is the (indirect) value to the customer of implementing the change?

   If there is no customer value to be found, even indirectly, or if a change request that is too similar already exists, the change request should be discarded. In this first step, it is important to keep the required formatting effort to a minimum to maintain a low threshold for taking in change requests. The field of direct beneficiary and customer value may be combined into a user story, which is a familiar concept to many engineers at T&I. Change requests reside in the system-level backlog. Until sufficient information is gathered in the change request, it will be labelled back and forth as “Needs refinement” and “refined” as the CCB decides more refinement is needed and the assigned person refines it, respectively.

2. Change requests are prioritized
   Using a requirements and feature prioritization strategy, the change requests are sorted. This results in an ordered lists of refined change requirements. If certain change requests cannot be fulfilled due to impediments, regardless of priority score, these should be marked as such to ensure reconsideration if the impediment is resolved.

3. Change requests are scheduled and distributed
   Starting with the requests with the highest priority, changes are scheduled for implementation, if possible. They are translated into subsystem requirements, which are linked to the original change request. These subsystem requirements are then transferred to their corresponding subsystem team.

4. Change requests are monitored and closed
   Scheduled change requests can be (temporarily) closed for three reasons: successful implementation, cancellation and impediment. Impediments must be monitored

more closely than reasons for cancellation. If an impediment is resolved, the change request can be resumed.

Azure DevOps is a suitable tool for continuous work, change, and issue tracking that the ICS team at Wavin T&I already has access to. However, DevOps is an extensive tool mainly aimed at software development, which is less relevant for system-level change request management. The interface elements related to software development can clutter up the UI, making the submission of change request possibly more strenuous. Nevertheless, DevOps allows for modification of the underlying schema, as well as which fields are shown for each work item. The change request workflow prescribed in the interactive handbook has been implemented by the systems architect in the ICS DevOps project on 06-04-2022.

Bug reporting can be seen as a subset (special case) of change requests. When reporting issues, three distinctions can be made. The first is a defect, which is a quality issue that arises when a product does not behave according to specification. The second is unintended behavior, which is when the customer thinks the product does not behave correctly when, in fact, it does work according to specification. The third issue category is bugs, which is when the product specification or functionality does not meet the requirements. All issues are applied in DevOps under the bug category, to facilitate issue reporting. If, after all, reporting an issue is a tedious and/or complicated process, the risk of them not being reported (properly) is introduced. The change control board filters bugs into two paths, depending on whether requirements need to be changed. If they do, the bug is closed, and a corresponding change request is created. Otherwise, the bug is prioritized using the same strategy for change requests, after which it is delegated to the relevant subsystem team. While a bug is resolved (and optionally after a bug-induced change request is implemented), creating a corresponding test case should be considered, in line with concurrent engineering principles [59].

For their Calefa DHW unit, Wavin T&I has recently implemented a change request process as part of a quality management improvement project. The proposed change request process from this methodology can be implemented in parallel to the Calefa change request process so long as Calefa remains a separate project from Sentio. In the future, if ICS projects are to be more integrated, overlapping quality management processes should be aligned or merged.

6.3.6 | Prioritization

Work items are a general term for actions in product development, such as the development of a feature. The problem of prioritizing work items is one of multicriteria as-

sessment. Having a defined prioritization strategy will help ICS at T&I strike a balance between quickly identifying and choosing work items to act upon first and spending time refining work items. Important in this multicriteria assessment is the required prioritization effort: no effort should be wasted defining and refining work items that might end up receiving a priority too low to further consider executing. This can be achieved by selecting the right assessment criteria, along with their granularity. Furthermore, an assessment method must be employed to prioritize the work items based on the chosen criteria. In this subsection, the development of a work item prioritization strategy is described.

6.3.6.1 | Criteria

Priority assessment aims to quantify the relative importance of items. Below, potential criteria of which a selection could indicate priority is provided, as well as their influence on priority:

- Business value: the magnitude of added value to the business [65]. A higher business value naturally receives a higher priority.
- Penalty: the adverse effects of not realizing the item [17, 136, 65]. An example of this is compromised market position. The presence of penalty for not implementing an item increases its priority.
- Investment: an estimation of the required costs of realization [17, 65]. A higher required investment lowers priority.
- Effort: the required amount of work, which can be seen as the combination of number of people and hours spent [17]. More difficult tasks indirectly increase investment and thus lower priority.
- System impact: the technical impact the realization has on other system components [65]. This indicates the risk of complications. Complications stagnate product development without adding business value, so work items with a higher technical impact should receive a lower priority.
- Urgency: a component of importance that indicates time-boundness [17]. A higher urgency may be conflated directly with a higher importance, but the distinction lies in the fact that urgency only indicates the time dependency. Nevertheless, a higher urgency does increase the priority.

- Kano model feature type: the comparison between the degree of implementation and the resulting customer satisfaction [2]. This can be seen as the combination between business value and penalty.
- Risk: the predicted risks associated with the implementation of the item [17, 65].
- Confidence: a component that originates from the RICE prioritization framework [104]. The RICE framework utilizes reach, impact, confidence, and effort. Impact and effort have already been covered, and reach is covered by business value. Confidence, however, indicates the certainty with which the overall priority assessment is made. This helps accompany for potential extreme scores based on gut feeling. A higher confidence score increases the priority of an item.

6.3.6.2 | Assessment methods

Below, an overview of relevant methods for that can be used for assigning priority is provided:

- Analytic hierarchy process: AHP is a systematic decision making method that involves pairwise comparison of all options. For larger numbers of requirements, this will result in too many comparisons [17, 85].
- QFD: quality function deployment inherently prioritizes potential features by assigning them to user needs [28, 85]. However, this requires a clear definition of user needs and technical implementation and does not consider the possibility to forego certain user needs.
- Cumulative voting (100 dollar test): cumulative voting is a prioritization method that works by constraining the number of points that can be assigned to each item, often illustrated as 100 dollars [17, 2, 60]. This analogy facilitates the prioritization process, as it familiarizes the problem.
- Numerical assignment: numerical assignment is a prioritization technique where requirements are divided between a limited number of predefined categories, which are assigned scores [17, 2, 60]. When performing numerical assignment, it is important to choose meaningful categories to reduce ambiguity and facilitate the process of assigning the requirements to the categories [17]. This technique is a quick, but very coarse method of prioritizing requirements.
- MoSCoW technique: the MoSCoW technique is very similar to numerical assignment, but has four fixed prioritization categories: must have, should have, could

have, won't have [2, 60]. It can be regarded as a direct combination between the Kano model and numerical assignment. In addition to being equally as inaccurate as numerical assignment, its support in prioritization is very limited, as defining in which category an item must go is actually the act of prioritization itself.

- Wiegers' matrix: when using Wiegers' prioritization approach, options are given a relative score, which is multiplied by a preset weight [136]. This prioritization method is fairly common [2] but suffers from the same limitations as QFD and AHP such as having a limited scale, not incorporating requirement dependencies, and not accounting for input subjectivity [85].
- Win-Win approach: the Win-Win approach is a fairly common [2] decision making framework that is focused on negotiation [18]. Its central principle is to maximize value for all involved stakeholders. As such, it is not a direct prioritization technique, but its central principle can be minded during the prioritization process.

6.3.6.3 | Strategy

If the number of criteria is too big, decision fatigue will arise. For every item, too many scores must then be decided upon. There are three remedies to reduce the number of decisions: reducing the number of criteria, reducing the granularity of some or all criteria, and providing default values for some or all criteria. Along with the criteria themselves and chosen assessment method, this comprises the work item prioritization strategy.

Since the scoring of work item criteria is a quick estimation, relative qualitative assessments are most suited. The approach of relative scales like in Wiegers' matrix fits best when the adaption is made of incorporating confidence as a criterion to correct for input subjectivity. Furthermore, the inclusion of technical impact as a criterion includes requirement dependencies in the prioritization process.

With relative scoring, a default value can be utilized, which can be tailored based on the estimation. For this reason, more criteria can be included, so long as the default values are selected carefully and not too many decisions must be made per item. The default value of all criteria is equal, starting with the base assumption that every work item is equally important. As criteria scoring is refined in the future, so is the quality of the overall priority assessment.

6.3.6.4 | Application

Development The initially selected criteria for Sentio development and their rationales are listed in table 6.2. Criteria that are not included in the prioritization strategy (such

Table 6.2: Initial prioritization criteria

Criterion	Rationale
Business value	Business value is the main driver of implementing a change.
Penalty	Penalties are useful for e.g. comparing fixing a bug with implementing a new feature.
Effort	Incorporating effort helps in identifying quick wins. The time estimate serves as a rough guideline; realistically, the number of people occupied may also be considered.
System impact	System impact is difficult to assess, especially with limited implemented traceability. However, a quick estimation helps to incorporate the wave of complications that may arise from implementing a certain feature.
Confidence	A confidence estimation can help temper unrealistic expectations in both directions.

as risk or estimated investment) can still be assessed when the work item is chosen for further investigation. If their values are deemed unacceptable, this can be reported as an impediment or reason for dismissal. The granularity and values of the scales must be adjusted over by evaluating the quality of the resulting prioritization. The selected criteria are used to calculate the relative priority score in the following first version of the formula, which can be seen as a loose adaptation of Wiegers' matrix, in which the weights are included in the numerical scales to facilitate implementation:

$$Priority = \frac{Business\ value * Penalty * Confidence}{Effort * System\ impact}$$

Change control board meeting The proposed strategy was discussed with the CCB (Project Manager Ronald Gorter and Systems Architect Stef Boerrigter) on 29-03-2022. The following conclusions were drawn:

- The penalty criterion should only be applied in one direction (1-2 instead of 0,25-2), as a penalty smaller than 1 does not carry any meaning.
- A rubric for all scores must be made to reduce personal variation in estimates. Additionally, the rubric for effort estimation should be changed from times in weeks to examples of which most engineers can get a sense of required effort.
- The business value estimation can only be made best by the business unit. The business value estimation should remain very rough and can be refined in later change request process steps.

Based on this meeting, the first version of the prioritization rubric was drafted, shown in table 6.3.

Table 6.3: Prioritization rubric, first version

Criterion	**Scale (scores)**	**Rubric**
Business value	Minor – small – normal – significant – huge (0,25 – 0,5 – 1 – 1,5 – 2)	Minor – Slight indirect customer value, e.g. improved code documentation Small – Indirect customer value or minor direct value, e.g. increased usage analytics or small QoL updates respectively Normal – Expected increase in sales, e.g. smart features such as presence detection Significant – Expected compliance with and exploitation of (future/local) regulation, e.g. CO2 monitoring or Sentio all-inclusive Huge – Game changer, serious step in establishing market leadership
Penalty	Normal – significant – huge (1 – 1,5 – 2)	Normal – Missed business value Significant – Compromised market position Huge – (Future/local) regulatory problems
Effort	Easy task – small task – normal task – difficult task – endeavor (0,25 – 0,5 – 1 – 1,5 – 2)	Easy task – Minor software feature e.g. Adding menu item in interface Normal task – Firmware feature e.g. protocol support Endeavor – Hardware revision or introduction of new component e.g. new PCB layout or adding CO2 sensor

System impact	Minor/nonexistent – limited – significant (0,5 – 1 – 1,5)	Minor/nonexistent – Minor software tweaks required, e.g. adding/changing menu items or less Limited – Software/firmware adaptation required or very minor hardware changes such as adding/changing a resistor Significant – Complete software or firmware revision required or large changes to hardware or mechanical design
Confidence	Uncertain – fairly certain – confident (0,5 – 1 – 1,5)	Uncertain – Multiple criteria may change with multiple points Certain – One criterion might need to be adjusted slightly

Version 1 was sent to the CCB for review and discussion. Changes were made based on extended discussion before applying it to a selection of current ICS change requests on 13-04-2022. Present at this exercise were Systems Architect Stef Boerrigter, Project Manager Ronald Gorter, and Product Manager Marco Oudshoorn. The following conclusions were drawn from this meeting and online post-meeting discussions:

- Wordings in the rubric, such as the reference to the CO_2 sensor and the wording for the 'huge' score at the penalty criteria should be changed.
- The terms 'easy' and 'difficult' at 'estimated effort' should be changed, as the current terms imply complexity, which is reserved for System impact.
- The rough time estimations at 'estimated effort' should be reintroduced, with endeavor requiring more than one year.
- System impact, in turn, should be renamed to technical impact. Furthermore, its focus on complexity and risk of complications should be emphasized in the rubric.
- Upfront, it was noticed that confidence scores influence the overall priority too much. For this reason, the impact of the confidence scores must be reduced.
- After evaluating the prioritized list of requests, it was concluded that the effort and system impact scores weigh too heavy.

- Urgency should be a criterion, indicating the time sensitivity of the change request.
- Urgency and effort can be interlinked by adapting the formula.
- Urgency should be a five-point scale to allow for more differentiation.

The revised formula is as follows:

$$Priority = Business\,value + Penalty - Urgency * Effort - Technical\,impact + Confidence$$

In table 6.4, version 2 of the rubric is provided. The scales have been adjusted to be in integers for easier implementation. Additionally, they have been normalized to zero instead of centered around neutral values to accommodate for the changed formula.

Table 6.4: Prioritization rubric, first version

Criterion	**Scale (scores)**	**Rubric**
Business value	Minor – small – normal – significant – huge (1 – 2 – 4 – 6 – 8)	Minor – Slight indirect customer value, e.g. improved code documentation Small – Indirect customer value or minor direct value, e.g. increased usage analytics or small QoL updates respectively Normal – Expected increase in sales, e.g. CO2 monitoring or smart features such as presence detection Significant – Expected compliance with and exploitation of (future/local) regulation, e.g. Sentio all-inclusive Huge – Game changer, serious step in establishing market leadership
Penalty	Normal – significant – huge (0 – 2 – 4)	Normal – Uncapitalized business value Significant – Compromised market position Huge – (Future/local) regulatory problems or a threat to the existence of the product

Urgency	Critical – urgent – normal – not urgent – time-indifferent (1 –2 – 3 – 4 – 5)	Critical – must be implemented within three months Urgent – must be implemented within six months Normal – must be implemented within one year Not urgent – must be implemented within two years Time-indifferent – can be implemented at any time
Effort	Quick task – small task – normal task – large task – endeavor (1 – 2 – 3 – 4 – 5)	Quick task – Within weeks; Minor software feature e.g. Adding menu item in interface Normal task – Within months; Firmware feature e.g. protocol support Endeavor – Within more than a year; Hardware revision or introduction of new component e.g. new PCB layout or adding CO2 sensor
Technical impact	Minor/nonexistent – Insignificant – Limited – Significant – Huge (0 – 1 – 2 – 3 – 4)	Minor/nonexistent – Minor software tweaks required, e.g. adding/changing menu items or less Limited – Software/firmware adaptation required or very minor hardware changes such as adding/changing a resistor Huge – Complete software or firmware revision required or large changes to hardware or mechanical design

Confidence	Uncertain – fairly certain – confident (0 – 1 – 2)	Uncertain – Multiple criteria may change with multiple points Certain – One criterion might need to be adjusted slightly

Implementation The prioritization strategy can be implemented into Azure DevOps by creating picklists with values for each of the criteria. Since DevOps does not allow for decimal values in picklists, integers must be used by multiplying the scale so that all values are whole numbers. Then, a Power Automate Workflow can read those values, divide them back to unit scale and insert them into the prioritization formula. If picklist values are changed, but the scale (more specifically the center of the scale) remains the same, only the affected work items need to be updated. If the center of the scale is changed, or if scales are added, removed, or renamed, the formula in the Power Automate Workflow must be updated as well. The prioritization process prescribed in the handbook (covered in section 6.6) using the aforementioned rubric has been successfully implemented in the ICS DevOps project.

6.3.7 | Contextualization and risk management

6.3.7.1 | Project context

The development approach of a product or system must be adapted to its context [37, 50]. To achieve this, contextual factors must be identified so that they can be incorporated in the product lifecycle. Du Preez et al. [37] distinguish between the following contextual characteristics:

- Project
  - size: the duration and amount of resources required for the project.
  - type: the nature of the product such as redesign, new development, or configuration design of existing components.
  - constraints: the limiting factors in the project such as budget and personnel.
  - complexity: the number of variables, number of (potentially conflicting) objectives, opacity, and interdependency all influence the risk and severity of complications.
- Organization

  - size: the size of the organization influences the formality and descriptiveness of the development process.
  - type: the identity of the organization such as their purpose, philosophies and strategies.
  - organizational maturity: the experience of the organization, especially related to product or system development
  - structure: the structure, position, and autonomy of the development team(s).
  - design capacity: the available resources for product development (or lack thereof) at the organization.

- Product
  - complexity: the number of subsystems and/or subproducts and their interrelatedness.
  - level within system hierarchy: indicates the number of interrelations with other products and (sub)systems.
  - type: can be divided in unit, batch or mass produced.

- Personnel
  - team size: the available human resources in the development project.
  - level of maturity: the experience and expertise of the project team.
  - design capability: can be seen as a combination between team size and maturity, influenced by management effectiveness, amounting to an overall design team competence.

The PESTLE framework can be used to classify these contextual factors. The PESTLE framework organizes Political, Economic, Social, Technological, and Environmental factors [105]. The aforementioned contextual factors span across all but the last of these. The PESTLE framework can be combined with the well-known SWOT (Strengths, Weaknesses, Opportunities, Threats) analysis [122], by placing the PESTLE factors on SWOT quadrants based on their influence and internality or externality, as illustrated in figure 6.3.

6.3.7.2 | Risk management

Project risk management is the process of identifying risks that may cause the project to deviate from the intended results and is directly related to project success [105]. The

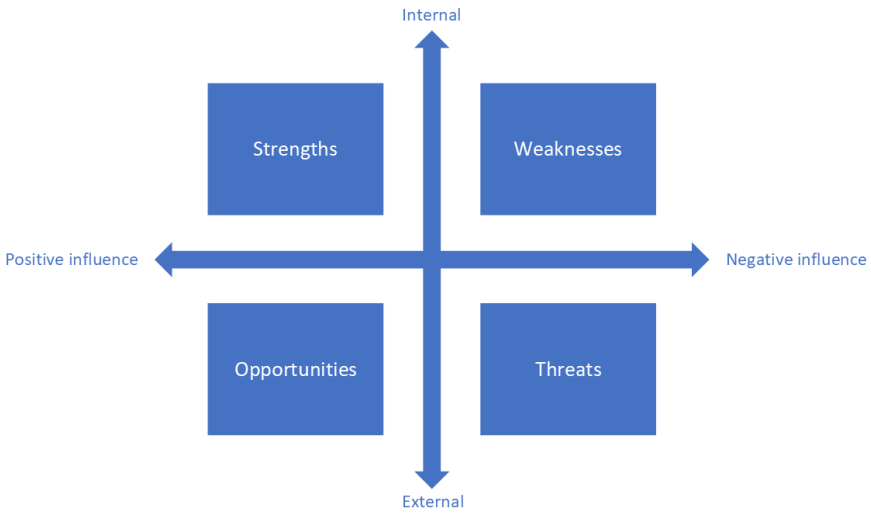


Figure 6.3: SWOT quadrants

Project Management Body of Knowledge prescribes the following risk management process [105]:

1. Plan risk management: define the risk management activities to be conducted.
2. Identify risks: generate an overview of all the project risks, as well as their origins.
3. Perform qualitative risk analysis: prioritize risks by assessing their impact and probability.
4. Perform quantitative risk analysis: quantify the effects and characteristics of the most relevant risks.
5. Plan risk responses: select a risk response strategy for the most relevant risks.
6. Implement risk responses: implement the selected strategy.
7. Monitor risks: monitor the implementation, its success, and the accuracy of the risk assessment.

A Failure Mode and Effects Analysis is a useful risk management tool that prioritizes potential product or system failures [19]. Performing an FMEA covers step 2 and 3 of the aforementioned risk management process.

6.3.7.3 | Application

The contextual factors can be stored in a standard scheme consisting of the combination of the PESTLE framework and the SWOT analysis as mentioned in paragraph 6.3.7.1, in order to successfully adapt the product development process to its context [50]. This context register provides the added benefit of easier familiarization of new project members to the development context. In addition to the identification and monitoring of contextual factors, risk assessment is a joint effort between the business unit and the project manager. By centralizing the storage, the resulting risk register and the contextual factors, as well as their requirements and constraints on the system design can be directly linked to it. This traceability ensures the tailoring of the design process to its context.

Matrix ALM allows for FMEA-style risk management. However, risks and contextual factors are applicable outside of their relation to (non-)functional requirements as well. Furthermore, not everyone may have access to Matrix ALM. Therefore, a central context and risk register would be best. These registers could, for example, also be referred to in impediments of work items. Currently, every project has a default folder structure in the central SharePoint of Wavin T&I. Risk assessments are placed in this project-bound folder as they are performed in an ad hoc fashion, causing them to be infrequently reused. For this reason, context and risk registers should be stored in a project-independent location.

6.4 | Metamodel

6.4.1 | Metamodels versus alternatives

As covered in subsection 6.3.1, metamodels are abstractions of models. A product development metamodel is, therefore, a model of the development process in general. This can be illustrated using the example of a recipe for a meal: the recipe itself can be seen as a process model of preparing the meal, whereas the metamodel of the recipe would be a model of the concepts in recipes in general, such as ingredients and instructions. Topic maps are similar in nature in the fact that they contain concepts and their relationships, but they are less strict and serve more of a communicative purpose. Ontologies have great potential use in PLM due to their support of automated reasoning. However, ontological developments are still facing too many limitations for practical application in PLM [72].

6.4.2 | Metamodels in product development

Metamodels formalize concepts, with the boundaries of the formalization being defined by the level of detail of the model. For example, expanding a metamodel beyond product development would include more and more of its organizational context, increasing the level of detail, eventually resulting in an enterprise architecture metamodel: a model of all building blocks used to define an enterprise architecture. The Open Group provides an enterprise architecture metamodel in their EA framework called TOGAF [127]. Logically, enterprise architectures and their metamodels far exceed product development in their scope. However, metamodels of product development exist, such as the Vitech product development metamodel [117]. This model has also been adapted to the development of defense systems [11]. Schön et al. [116] have developed a metamodel for agile requirements engineering, which is more oriented on the process of development rather than the definition of the system. Carniel and Pegoraro [26] propose a metamodel for the traceability of releases to their user stories. These metamodels all provide useful perspectives on the concepts in product development, and form the basis of the product development metamodel for Wavin T&I.

6.4.3 | Structure

By assessing which of the concepts in existing product development metamodels also apply to Sentio development, the basic components of the metamodel are identified. When defining their relationships, the boundary of formalization must be chosen. In the interest of knowledge management practices, as well as in the interest of traceability, the boundary is drawn at decisions, investigations, risks and contextual factors. The resulting metamodel is provided in figure 6.4.

The decision to combine common system concepts such as ‘state’, ‘component’, and ‘interface’ into a single ‘design’ concept has been made actively to maintain a process perspective over a system definition perspective. The system architecture should define these contents of the design. The metamodel has been colored for illustrative purposes. The model items in dark blue define the intention and result of the system design. The lighter shades of blue indicate verification and validation activities, their execution, and their results. Lastly, the items marked in gray are situational influences on the system definition.

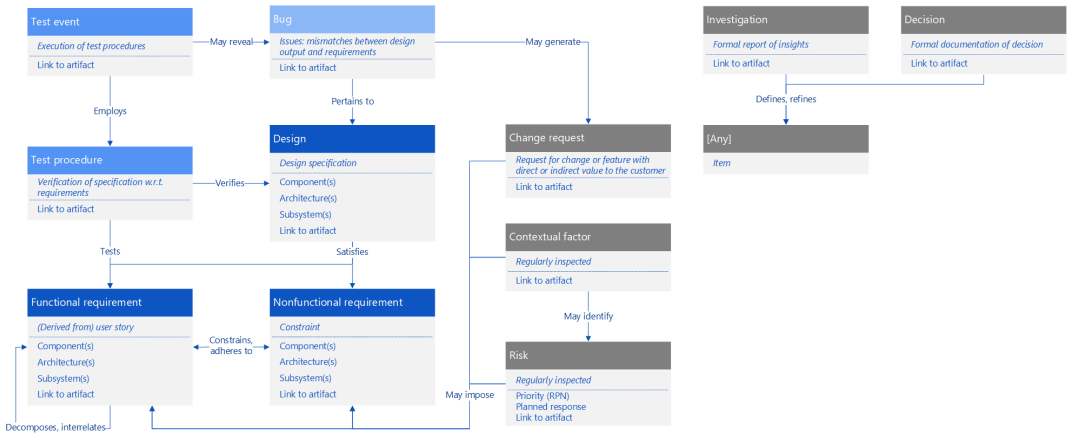


Figure 6.4: System development metamodel (enlarged in Appendix E)

6.5 | Principles

As mentioned in section 5.1.2, principles bind a set of methods and tools to form a methodology. The principles can be common denominators across methods, principles from popular existing methodologies and principles derived from business strategy and values. In this section, the principles that bind the methodology are covered. They serve as general guidelines when implementing the methodology.

Transparency Transparency means the access of as many stakeholders as possible to as much information as possible. Open innovation requires trustful, sustainable development partnerships and has been shown to improve knowledge management capacity [112]. Furthermore, knowledge sharing increases customer value and reduces product development lead times [58]. Transparency is also one of the core values in the Scaled Agile framework [113].

Bottom-up approach A bottom-up development approach means utilizing the technical expertise and market knowledge of development engineers as much as possible. This is necessary to capitalize on rapidly arising opportunities [125].

Decision decentralization To realize the servitization ambitions, Wavin should guide product development from a service perspective. ICS services then deliver value through software as a medium, with hardware being the medium outlet. In product-service system oriented business creation, as opposed to the traditional product-oriented business creation, the business is involved in value creation along the entire lifecycle of the prod-

uct [125]. The direction of this service should be determined by senior management, guided by the PSS philosophy. The system-level direction should be determined at an architectural level, using, for example, the ternary plot from the Sentio 2.0 section of the background chapter. The technical direction of the subsystems should be determined as much as possible by the relevant subsystem teams. This leads to the principle of decision decentralization [113]. This principle is supported by transparency and a bottom-up approach.

Single source of truth To facilitate knowledge management, a single source of truth must be pursued. Successful implementation of a single source of truth reduces misunderstandings caused by outdated and/or duplicate information.

Traceability Traceability not only supports quality management as mentioned in the document and configuration management section, but also documents reasoning by allowing for answers to be traced to their questions and vice versa.

Scalability As the gaps the methodology aims to address originate from issues of scale, scalability must be held in high regard. This is facilitated by the decision decentralization principle, as well as principle 11 of the Agile Manifesto: “the best architectures, requirements, and designs emerge from self-organizing teams” [16].

Continuous improvement In line with principle 12 of the Agile Manifesto, regular reflection and improvement is essential [16]. The methodology and system development in general must be frequently evaluated as the ICS venture evolves.

6.6 | Implementation in Wavin T&I

6.6.1 | Prescriptions

The PLM methodology is applied to Wavin T&I using a set of prescriptions on the following eight topics:

- Document and configuration management
- Decision and investigation management
- Feature request processing
- Bug processing

- Prioritization
- System requirements management
- Risk management and contextualization
- Verification and validation

Combined, these topics apply all of the methodology components. Furthermore, they each cover an area of the metamodel. The prescriptions provide instructions on how the methods from the methodology should be applied, using which tools. The prescriptions contain sections on implementation in the software already present at Wavin T&I: Azure DevOps, Matrix ALM, and Microsoft SharePoint. This decision has been made over the implementation of a full PLM suite for the purpose of direct applicability and scale: the procurement and implementation of a PLM suite would be an enormous endeavor resulting in a solution that is excessive for the current scale of system development.

6.6.2 | Handbook

The prescriptions have been presented in an interactive handbook in the form of a website. This website is hosted internally, granting access only to those who may be confided in its contents. Due to the many interrelations between the topics of the prescription, interactivity of the handbook was deemed a necessity. The two prime candidates for interactive handbook technology were PDF based on a PowerPoint, and a website. Drawbacks of a website are maintainability and required effort, whereas its benefits are accessibility and flexibility. The benefit of an interactive PDF is its simplicity, but the fact that it must be shared as a file through services such as e-mail or SharePoint poses significant threats for maintaining a single up-to-date version at everybody's disposal. Furthermore, PDF interactivity is extremely limited. The benefits of developing a website have been deemed outweighing of its drawbacks, in part thanks to the application of a markdown-based static site generator named Hugo[1]. This technique reduces the difficulty and required web-development knowledge for making adaptations to the handbook, by generating the page from more legible markdown files. The central component of the website is the metamodel, which serves as a map of all the prescription topics, as shown in figure 6.5. By hovering over the buttons for each of the topics, their relevant area in the metamodel is highlighted.

Every topic has its own page describing the prescription, of which an example is shown in figure 6.6. References to principles and other topics are formatted as hyperlinks, allow-

[1] https://gohugo.io

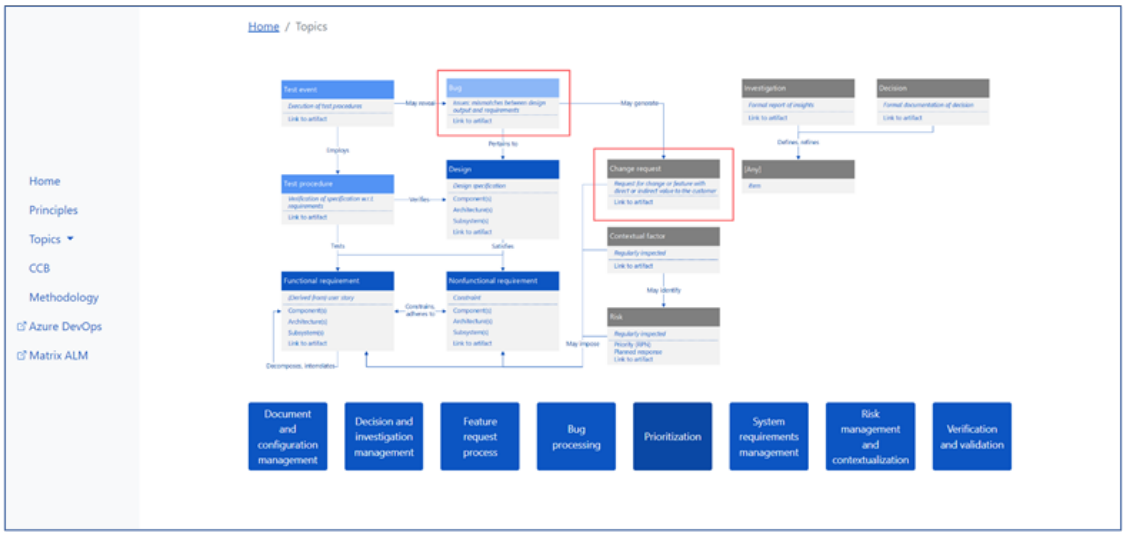


Figure 6.5: Interactive handbook

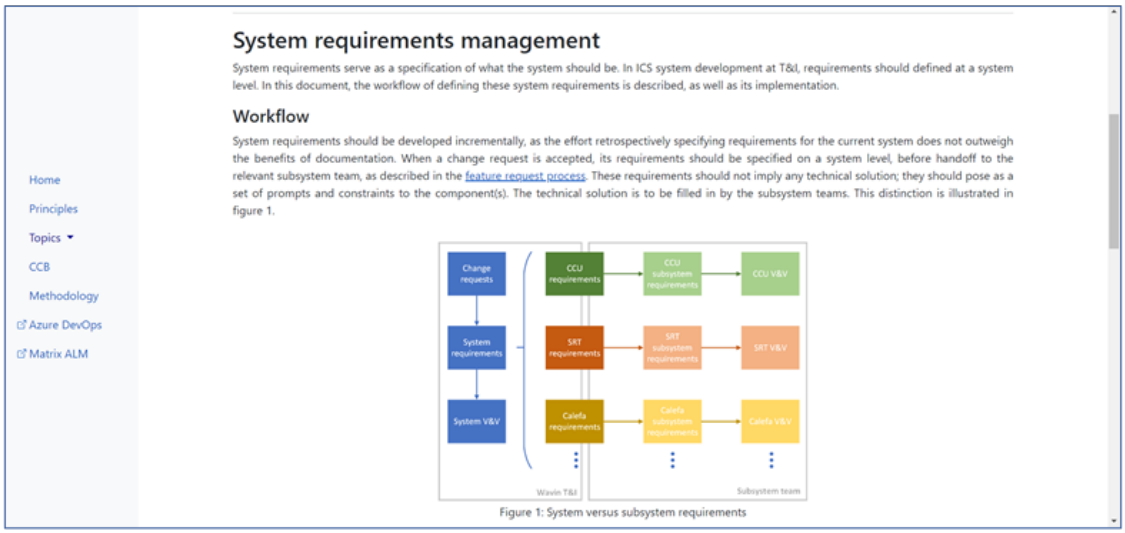


Figure 6.6: One of the topic pages

ing the reader to quickly navigate back and forth between topics. Furthermore, general pages on the underlying principles, the CCB, and the methodology itself are included. Finally, quick links to the ICS DevOps and Matrix ALM environments have been provided.

The purpose of the interactive handbook is to provide a quick reference for all PLM related developments. It aims to increase support for new product development management tools and processes by providing a rationale. Furthermore, it is supposed to be an open-ended document, and is to be read out of interest rather than obligation.

6.7 | Roadmap

The application of the methodology is fully dependent on the scale of system development at Wavin T&I. As the ICS venture expands, PLM efforts should be scaled up accordingly. The first improvement beyond the implementation of the handbook prescriptions is the alignment of PDM and version control systems across stakeholders. Furthermore, SharePoint practices could be improved by creating dedicated libraries for definitive releases, as well as departments in general. This will allow for finer control of access and automation and make project folders and files more manageable.

The prescriptions for the current PLM toolset (DevOps, Matrix and SharePoint) will reach its limitations at a certain scale of product development. It is at this point that a PLM software suite should be considered to support information management. If a significant amount of time is spent servicing the PLM toolset, it is a sign that this transition point is approaching. The benefits of a PLM suite will then outweigh the high upfront investment of its expensive license and complicated rollout.

If a PLM suite is commissioned, the gateway to model-based systems engineering is opened. MBSE will enable for complex system simulation testing and automation. If further formalization of the ICS enterprise is desired, MBSE and the methodology metamodel can be adapted in an enterprise architecture. In the far future, developments in ontology and ontology based systems engineering may further support PLM by, for instance, providing relevant information based on semantic similarity or applying automated reasoning to find logical inconsistencies in the system.

7

Discussion

In order to establish the extent to which the methodology generalizes to other companies, the characteristics of the development situation are discussed. Then, the aspects that influence how the methodology should be applied are covered. Finally, critical remarks on PLM maturity assessment are provided, as well as an explanation on how the methodology can address its shortcomings.

7.1 | Characteristics

The situation at Wavin T&I Sentio development is characterized by mainly by its relative novelty within the established company. As developing a smart product that consists of both hardware and software was not the expertise of Wavin, the benefits of outsourcing the development of at first the AHC-9000 and later Sentio far outweighed the costs of developing the products in-house. This outsourcing, however, brings the effect of reduced project ownership with it: the products are engineered for Wavin, rather than by Wavin. Correspondingly, information management practices are fully tailored to product development practices of the main products of the company. Other companies looking to venture into new markets will face the same issues, depending on the novelty of the developments in question. If the business has (acquired) expertise in the development of the new products already, the reduction of project ownership in outsourcing will be less significant. As mentioned in the section 6.7, a certain degree of project convolution will require a more sophisticated PLM toolset, such as a complete PLM software suite. PLM software vendors present large manufacturers of complex products and systems as clients on their websites, indicating their recognition of the need of PLM toolsets. Still, the complexity of rollout and lengthy procurement process of these predominant PLM

suites shows that there are no out-of-the-box solutions. The PLM methodology of this thesis aims to bridge this gap.

7.2 | Adaptations

To apply the PLM methodology to other organizations, adaptations must be made. The considerations of applying the PLM methodology are structured using the contextual factors of Du Preez et al. [37], introduced in the risk management and contextualization component (paragraph 6.3.7.1).

- Project
  - size: in very small projects, the benefits of implementing the PLM methodology may not outweigh the effort. In very large projects, a full PLM suite may be more suitable.
  - type: development of new products may present the opportunity to implement new PLM practices, whereas a redesign or small design revision may be disrupted too much.
  - constraints: budget for PLM tools and personnel to manage product information naturally limits the extent to which the methodology can be applied.
  - complexity: highly complex projects benefit more from robust PLM implementation.
- Organization
  - size: in larger organizations, on the one hand, the process of implementing a PLM methodology may be more complicated and take longer. On the other hand, it can be coordinated and applied across multiple projects, and even departments, increasing its benefits.
  - type: the strategy of the organization significantly influences the applicability of the methodology. If the organization does not intend to scale up development efforts, applying a methodology aimed at facilitating growth is useless.
  - organizational maturity: if a company already has experience in developing more complex products, they may already have sufficient information management practices in place. Still, rigidity in the development process may be a challenge the PLM methodology can resolve.
  - structure: in companies with a strict hierarchy, the principles of the bottom-up approach and decision decentralization may not be desired or applicable.

  - design capacity: if a company has all design capacity in-house already, the benefits of the methodology can increase, especially if there are no existing PLM practices in place.

- Product
  - complexity: management of information for uncomplicated products is not a challenge and applying a methodology might then be excessive. Additionally, some products are held to such high safety and quality standards or may be of such complexity, as in the medical device or aerospace industry respectively, that the implementation of far more rigid and formalized information management processes are required at all scales of product development.
  - level within system hierarchy: if product development is dependent on much information of external components, the management of contextual information becomes more prominent in the implementation of PLM practices.
  - type: the scale of manufacturing from unit to batch, to mass production is often correlated with the product complexity. For complex systems sold in units, information management becomes more important in the later stages of the lifecycle for maintainability, for instance. For mass produced products it may be more important in the manufacturing lifecycle stage for quality assurance.
- Personnel
  - team size: the number of project members, as well as their degree of distribution influences the characteristics of knowledge exchange. Larger teams working from multiple locations need to externalize information more frequently, as opposed to direct exchange in conversations.
  - level of maturity: the prior experience of project members influences their knowledge and receptivity of PLM techniques.
  - design capability: design capability influences the time dependence of the application of the methodology. If the development team is rapidly generating information, its management becomes more critical.

Besides these contextual factors indicating the need and applicability of PLM, PLM maturity in general also influences the applicability of the methodology. Companies may already be practicing PLM unknowingly, for example. This would reduce the applicability of the methodology, as its methods or similar methods are already applied. Still, its educative function would still be useful in strengthening the PLM mindset of the company.

This mindset can improve the reception of tools, methods, and processes, which could otherwise be seen as bureaucratic.

7.3 | PLM maturity

Even though the areas that common PLM maturity models assess apply to the identified gaps and the components of the PLM methodology [97, 15, 101, 120], they provide no suggestion on improvement of these areas. The aim of PLM maturity assessment is to structure and facilitate the implementation of PLM [134]. However, maturity assessment is only useful when paired with suggestions, or at least considerations, to enable improvement. Concededly, PLM maturity and PLM maturity assessment is a 'chicken and egg' situation, as the same knowledge and experience required to assess PLM maturity effectively and draw useful conclusions from it is used for PLM implementation and improving maturity in the first place. In the future, PLM maturity assessment can certainly be a valuable addition to a PLM methodology, to steer implementation efforts.

8

Conclusion

Wavin T&I intends to scale up development management efforts for the second generation of their Sentio product line. To achieve this, improved information management practices across the lifecycles of both generations are required. This study has aimed to address the issue as follows. By investigating relevant fields, methodologies, methods, models, and tools for PSS development in an explorative analysis, an overview of relevant concepts was created. These have been applied to Wavin T&I development in the form of a methodology that addresses the information management gaps using methods and principles developed and curated from the findings of the explorative analysis. This methodology is structured around a system development metamodel, which identifies the types of information involved in development. The methodology was developed in a bidirectional approach, bringing academic work into practice in existing and available processes and software tools. Besides the direct implementation of methodology components in T&I development, implementation prescriptions have been processed into an interactive handbook in the form of a website. This approach immediately brings PLM concepts into practice, providing a stepping stone to larger commercial PLM solutions in the future.

Further research can investigate the transition from initial PLM practices as prescribed by this methodology to model or even ontology based systems engineering, as well as the incorporation of PLM maturity assessment. Moreover, future works can further determine the signals that a custom PLM solution of existing tools must be superseded by larger commercial PLM solutions. Overall, this research has pointed out that while PLM is a useful paradigm in managing the information of product development, its broad scope can limit its direct application. The developed methodology can help companies partaking in new and growing development endeavors overcome this challenge.

References

[1] Quality System Regulation. Regulation 21 CFR 820, Code of Federal Regulations. United States Federal Government, Mar. 2020.

[2] P. Achimugu, A. Selamat, R. Ibrahim, and M. N. R. Mahrin. A systematic literature review of software requirements prioritization research. *Information and Software Technology*, 56:568–585, 2014.

[3] R. Anderl, K. Mecke, A. Sprenger, and O. Weitzmann. Ontology support for product development - successful application of ontologies in product development. pages 177–182, 01 2009.

[4] M. M. Andreasen. 45 years with design methodology. *Journal of Engineering Design*, 22:293–332, 5 2011.

[5] P. O. Antonino, T. Keuler, N. Germann, and B. Cronauer. A non-invasive approach to trace architecture design, requirements specification and agile artifacts. In *Proceedings of the Australian Software Engineering Conference, ASWEC*, pages 220–229. IEEE Computer Society, 2014.

[6] S. Ariwaka, H. Nakagawa, and T. Tsuchiya. Graph queries for analyzing the coverage of requirements by test cases. In *Proceedings of the International Conference on Software Engineering and Knowledge Engineering, SEKE*, volume 2021-July, pages 544–549. Knowledge Systems Institute Graduate School, 2021.

[7] H. Arnold, M. Erner, P. Möckel, and C. Schläffer. *Enterprise Architecture in Innovation Implementation*, pages 132–144. Springer Berlin Heidelberg, 2010.

[8] Association for Intelligent Information Management. What is ECM? `https://info.aiim.org/what-is-ecm`. Accessed: 2022-05-29.

[9] R. Atkinson. Project management: cost, time and quality, two best guesses and a phenomenon, its time to accept other success criteria. *International Journal of Project Management*, 17:337–342, 12 1999.

[10] Atlassian. Gitflow workflow. `https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow`. Accessed: 2022-05-29.

[11] G. Bakirtzis, T. Sherburne, S. Adams, B. M. Horowitz, P. A. Beling, and C. H. Fleming. An ontological metamodel for cyber-physical system safety, security, and resilience coengineering. *Software and Systems Modeling*, 21:113–137, 2 2022.

[12] J. Balogun and V. Hope-Hailey. *Exploring Strategic Change (2nd ed.)*. Prentice-Hall, 2003.

[13] R. Bandinelli, E. d'Avolio, M. Rossi, S. Terzi, and R. Rinaldi. Assessing the role of knowledge management in the new product development process: An empirical study. In S. Fukuda, A. Bernard, B. Gurumoorthy, and A. Bouras, editors, *Product Lifecycle Management for a Global Market*, pages 397–406, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[14] M. J. Barrenechea and T. Jenkin. *Enterprise Information Management: The Next Generation of Enterprise Software*. More Than Words, 2013.

[15] R. Batenburg, R. W. Helms, and J. Versendaal. PLM roadmap: stepwise PLM implementation based on the concepts of maturity and alignment. *Int. J. Product Lifecycle Management*, 1:333–351, 2006.

[16] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. Mellor, K. Schwaber, J. Sutherland, and D. Thomas. Manifesto for agile software development. `https://agilemanifesto.org`, 2001. Accessed: 2022-05-29.

[17] P. Berander and A. Andrews. *Requirements Prioritization*, pages 69–94. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[18] B. Boehm and H. Kitapci. *The WinWin approach: Using a requirements negotiation tool for rationale capture and use*, pages 173–190. Springer Berlin Heidelberg, 2006.

[19] G. M. Bonnema, K. T. Veenvliet, and J. F. Broenink. *Systems design and engineering : facilitating multidisciplinary development projects*. CRC Press, 2015.

[20] G. Boothroyd. *Design for Manufacture and Assembly: The Boothroyd-Dewhurst Experience*, pages 19–40. Springer Netherlands, Dordrecht, 1996.

[21] K. C. Bourne. *Application Administrators Handbook*. Morgan Kaufmann, 2014.

[22] D. J. Bradfield and J. X. Gao. A methodology to facilitate knowledge sharing in the new product development process. *International Journal of Production Research*, 45(7):1489–1504, 2007.

[23] J.-S. Brunner, L. Ma, C. Wang, L. Zhang, D. C. Wolfson, Y. Pan, and K. Srinivas. Explorations in the use of semantic web technologies for product information management. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, page 747–756, New York, NY, USA, 2007. Association for Computing Machinery.

[24] R. T. By. Organisational change management: A critical review. *Journal of Change Management*, 5:369–380, 2005.

[25] P. Canonico, E. D. Nito, V. Esposito, G. Fattoruso, M. P. Iacono, and G. Mangia. Visualizing knowledge for decision-making in lean production development settings. insights from the automotive industry. *Management Decision*, 60:1076–1094, 3 2022.

[26] C. A. Carniel and R. A. Pegoraro. Metamodel for requirements traceability and impact analysis on agile methods. volume 802, pages 105–117. Springer Verlag, 2018.

[27] Y. C. Cavalcanti, P. A. da Mota Silveira Neto, I. d. C. Machado, T. F. Vale, E. S. de Almeida, and S. R. d. L. Meira. Challenges and opportunities for software change request repositories: a systematic mapping study. *Journal of Software: Evolution and Process*, 26(7):620–653, 2014.

[28] L. K. Chan and M. L. Wu. Quality function deployment: A comprehensive review of its concepts and methods. *Quality Engineering*, 15:23–35, 9 2002.

[29] B. Chandrasekaran, J. R. Josephson, and R. Benjamins. What are ontologies, and why do we need them? *IEEE Intell. Syst.*, 14:20–26, 1999.

[30] U. Cugini, A. Ramelli, C. Rizzi, and M. Ugolotti. *Total Quality Management and Process Modeling for PLM in SME*, pages 339–350. Springer London, London, 2006.

[31] N. Dedić. Eafp: Enterprise architecture fusion process. *Journal of Information and Organizational Sciences*, 45(1), Jun. 2021.

[32] Design Council. A study of the design process. `https://www.designcouncil.org.uk/sites/default/files/asset/document/ElevenLessons_Design_Council%20(2).pdf`, 2005. Accessed: 2022-05-29.

[33] H. Do and G. Rothermel. An empirical study of regression testing techniques incorporating context and lifetime factors and improved cost-benefit models. 2006.

[34] B. P. Douglass. Chapter 2 - what are agile methods and why should i care? In *Agile Systems Engineering*, pages 41–84. Morgan Kaufmann, Boston, 2016.

[35] A. Dreibelbis, E. Hechler, I. Milman, M. Oberhofer, P. van Run, and D. Wolfson. *Enterprise Master Data Management: An SOA Approach to Managing Core Information*. IBM Press, 01 2008.

[36] V. Driessen. A succesful Git branching model. `https://nvie.com/posts/a-successful-git-branching-model/`, 2010. Accessed: 2022-05-29.

[37] N. du Preez, D. Lutters, and H. Nieberding. Tailoring the development process according to the context of the project. *CIRP Journal of Manufacturing Science and Technology*, 1:191–198, 2009.

[38] enterprise-information-management.com. How engineering information management systems complement PLM. `https://www.engineering-information-management.com/how-plm-only-manages-50percent-of-engineering-data-and-eim-complements-it/`. Accessed: 2022-05-29.

[39] European Commission. Electromagnetic Compatibility (EMC) Directive. `https://ec.europa.eu/growth/sectors/electrical-and-electronic-engineering-industries-eei/electromagnetic-compatibility-emc-directive_en`, 2014. Accessed: 2022-05-29.

[40] European Commission. Low Voltage Directive (LVD). `https://ec.europa.eu/growth/sectors/electrical-and-electronic-engineering-industries-eei/low-voltage-directive-lvd_en`, 2014. Accessed: 2022-05-29.

[41] European Commission. Radio Equipment Directive (RED). https://ec.europa.eu/growth/sectors/electrical-and-electronic-engineering-industries-eei/radio-equipment-directive-red_en, 2014. Accessed: 2022-05-29.

[42] European Union. CE marking. https://europa.eu/youreurope/business/product-requirements/labels-markings/ce-marking/index_en.htm. Accessed: 2022-05-29.

[43] G. D. Everett and R. Mcleod. *Software Testing Testing Across the Entire Software Development Life Cycle*. Wiley-IEEE Press, 2007.

[44] R. K. Faris, K. T. Neckowicz, and K. Isfahani. Enterprise project portfolio management: a must for project-based success. Project Management Insitute, 2 2010.

[45] C. Favi, M. Germani, and M. Mandolini. Design for manufacturing and assembly vs. design to cost: Toward a multi-objective approach for decision-making strategies during conceptual design of complex products. In *Procedia CIRP*, volume 50, pages 275–280. Elsevier B.V., 2016.

[46] P. Fitsilis, V. Gerogiannis, and L. Anthopoulos. Ontologies for project management: Survey. *International Journal of Information Processing and Management*, 5:1–7, 11 2014.

[47] J. Gao and A. Bernard. An overview of knowledge sharing in new product development. *International Journal of Advanced Manufacturing Technology*, 94:1545–1550, 2 2018.

[48] Gartner. Definition of innovation management. https://www.gartner.com/en/information-technology/glossary/innovation-management. Accessed: 2022-05-29.

[49] C. Gellweiler. Connecting enterprise architecture and project portfolio management: A review and a model for it project alignment. *International Journal of Information Technology Project Management*, 11:99–114, 1 2020.

[50] K. Gericke, M. Meißner, K. Paetzold, and I. K. Gericke. Understanding the context of product development. In *International Conference on Engineering Design*. Sungkynkwan University, 8 2013.

[51] M. Glinz. On non-functional requirements. *15th IEEE International Requirements Engineering Conference (RE 2007)*, pages 21–26, 2007.

[52] Google Trends. plm - Interest over time. https://trends.google.com/trends/explore?date=all&q=plm. Accessed: 2022-05-29.

[53] Google Trends. web 2.0 - Interest over time. https://trends.google.com/trends/explore?date=2005-04-16%202022-05-16&q=web%202.0. Accessed: 2022-05-29.

[54] K. R. Grahlmann, R. W. Helms, C. Hilhorst, S. Brinkkemper, and S. V. Amerongen. Reviewing enterprise content management: A functional framework. *European Journal of Information Systems*, 21:268–286, 2012.

[55] H. Graves. Ontology engineering for product development. http://ceur-ws.org/Vol-258/paper02.pdf. Accessed: 2022-05-29.

[56] P. Hammant. Trunk based development: Introduction. `https://trunkbaseddevelopment.com/`. Accessed: 2022-05-29.

[57] M. Hoffmann, N. Kühn, M. Weber, and M. Bittner. Requirements for requirements management tools. In *Proceedings of the IEEE International Conference on Requirements Engineering*, pages 301–308, 2004.

[58] P. Hong, W. J. Doll, A. Y. Nahm, and X. li. Knowledge sharing in integrated product development. *European Journal of Innovation Management*, 7:102–112, 6 2004.

[59] J. Hoppmann, E. Rebentisch, U. Dombrowski, and T. Zahn. A framework for organizing lean product development. *EMJ - Engineering Management Journal*, 23:3–15, 3 2011.

[60] A. Hudaib, R. Masadeh, M. H. Qasem, and A. Alzaqebah. Requirements prioritization techniques comparison. *Modern Applied Science*, 12:62, 1 2018.

[61] T. Huldt and I. Stenius. State-of-practice survey of model-based systems engineering. *Systems Engineering*, 22:134–145, 3 2019.

[62] Functional safety of electrical/electronic/programmable electronic safety-related systems - Part 1: General requirements. Standard 61508-1:2010, International Electrotechnical Commission, Geneva, CH, Apr. 2010.

[63] I. M. Ilevbare, D. Probert, and R. Phaal. A review of TRIZ, and its benefits and challenges in practice. *Technovation*, 33:30–37, 2 2013.

[64] International Council on Systems Engineering. Systems engineering. `https://www.incose.org/systems-engineering`. Accessed: 2022-05-29.

[65] Quality management — Guidelines for configuration management. Standard 10007:2017, International Organization for Standardization, Geneva, CH, Mar. 2017.

[66] Information technology — Topic Maps — Part 5: Reference model. Standard 13250-5:2015, International Organization for Standardization, Geneva, CH, Apr. 2015.

[67] Medical devices — Quality management systems — Requirements for regulatory purposes. Standard 13485:2016, International Organization for Standardization, Geneva, CH, Mar. 2016.

[68] Information technology — Object management group systems modeling language (OMG SysML). Standard 19514:2017, International Organization for Standardization, Geneva, CH, Mar. 2017.

[69] Road vehicles — Functional safety — Part 1: Vocabulary. Standard 26262-1:2011, International Organization for Standardization, Geneva, CH, Nov. 2011.

[70] Innovation management — Fundamentals and vocabulary. Standard 56000:2020, International Organization for Standardization, Geneva, CH, Feb. 2020.

[71] Quality management systems — Requirements. Standard 9001:2015, International Organization for Standardization, Geneva, CH, Mar. 2015.

[72] S. E. Kadiri and D. Kiritsis. Ontologies in the context of product lifecycle management: State of the art literature review. *International Journal of Production Research*, 53:5657–5668, 9 2015.

[73] U. Kampffmeyer. *Enterprise Content Management*. PROJECT CONSULT Unternehmensberatung Dr. Ulrich Kampffmeyer GmbH, 2006.

[74] N. Kano, N. Seraku, F. Takahashi, and S. Tsuji. Attractive quality and must-be quality. *The Journal of the Japanese Society for Quality Control*, 14:39–44, 01 1984.

[75] M. E. Khan and F. Khan. A comparative study of white box, black box and grey box testing techniques. *IJACSA) International Journal of Advanced Computer Science and Applications*, 3, 2012.

[76] W. C. Kim and R. Mauborgne. *Blue Ocean Strategy*. Harvard Business Review Press, 2015.

[77] J. Kuchta. Completeness and consistency of the system requirement specification. In *Position Papers of the 2016 Federated Conference on Computer Science and Information Systems*, volume 9, pages 265–269. PTI, 10 2016.

[78] K. Kulkarni, V. N. Kulkarni, V. N. Gaitonde, and B. B. Kottur shettar. State of the art review on implementation of product lifecycle management in manufacturing and service industries. In *AIP Conference Proceedings*, volume 2316. American Institute of Physics Inc., 2 2021.

[79] E. Landre, H. Wesenberg, and H. Rønneberg. Architectural improvement by use of strategic level domain-driven design. In *OOPSLA 2006: 21st Intenational Conference on Object-Oriented Programming, Systems, Languages, and Applications*, pages 809–814, 01 2006.

[80] R. Lööf and K. Pussinen. Visualisation of requirements and their relations in embedded systems. Master thesis, Uppsala University, 2014.

[81] A. M. Madni and M. Sievers. Model-based systems engineering: Motivation, current status, and research opportunities. *Systems Engineering*, 21:172–190, 5 2018.

[82] P. G. Maropoulos and D. Ceglarek. Design verification and validation in product lifecycle. *CIRP Annals - Manufacturing Technology*, 59:740–759, 2010.

[83] A. Martakis and M. Daneva. Handling requirements dependencies in agile projects: A focus group with agile software development practitioners. In *Proceedings - International Conference on Research Challenges in Information Science*, 2013.

[84] A. L. Mendelow. Environmental scanning - the impact of the stakeholder concept. In *International Conference on Information Systems*. Association for Information Systems, 1981.

[85] F. Moisiadis. The fundamentals of prioritising requirements. In *Systems Engineering, Test & Evaluation Conference*, 10 2002.

[86] Y. Mordecai and D. Dori. Model-based requirements engineering: Architecting for system requirements with stakeholders in mind. *2017 IEEE International Systems Engineering Symposium (ISSE)*, pages 1–8, 2017.

[87] B. Murphy, A. Wakefield, and J. Friedman. Best practices for verification, validation, and test in model-based design. In *SAE World Congress & Exhibition*, 4 2008.

[88] B. Nadia, G. Gregory, and T. Vince. Engineering change request management in a new product development process. *European Journal of Innovation Management*, 9:5–19, 1 2006.

[89] C. Nakata. Design thinking for innovation: Considering distinctions, fit, and use in firms. *Business Horizons*, 63:763–772, 11 2020.

[90] Y. Nemoto, F. Akasaka, and Y. Shimomura. A framework for managing and utilizing product-service system design knowledge. *Production Planning and Control*, 26:1278–1289, 11 2015.

[91] P. K. Ng, G. Goh, and U. Eze. The role of knowledge management in product development performance: A review. *Journal of Knowledge Management Practice*, 12, 03 2011.

[92] E. W. Ngai and E. W. Chan. Evaluation of knowledge management tools using ahp. *Expert Systems with Applications*, 29:889–899, 11 2005.

[93] S. Nidhra. Black box and white box testing techniques - a literature review. *International Journal of Embedded Systems and Applications*, 2:29–50, 6 2012.

[94] A. Nilsson, J. Bosch, and C. Berger. Visualizing testing activities to support continuous integration: A multiple case study. In G. Cantone and M. Marchesi, editors, *Agile Processes in Software Engineering and Extreme Programming*, pages 171–186, Cham, 2014. Springer International Publishing.

[95] I. Nonaka and H. Takeuchi. *The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation*. Oxford University Press, 1995.

[96] B. W. Oppenheim. Lean product development flow. *Systems Engineering*, 7, 2004.

[97] M. Paavel, K. Karjust, and J. Majak. Plm maturity model development and implementation in sme. In *Procedia CIRP*, volume 63, pages 651–657. Elsevier B.V., 2017.

[98] G. Pahl and W. Beitz. *Konstruktionslehre*. Springer Berlin Heidelberg, 1997.

[99] D. Pandey, U. Suman, and A. K. Ramani. An effective requirement engineering process model for software development and requirements management. In *Proceedings - 2nd International Conference on Advances in Recent Technologies in Communication and Computing, ARTCom 2010*, pages 287–291, 2010.

[100] P. Patanakul, B. Iewwongcharoen, and D. Milosevic. An empirical study on the use of project management tools and techniques across project life-cycle and their impact on project success. *Journal of General Management*, 35, 2010.

[101] H. J. Pels and K. Simons. *PLM Maturity Assessment*, pages 645–652. University of Nottingham, 2008.

[102] L. Plonka, H. Sharp, P. Gregory, and K. Taylor. UX design in agile: A DSDM case study. In G. Cantone and M. Marchesi, editors, *Agile Processes in Software Engineering and Extreme Programming*, pages 1–15, Cham, 2014. Springer International Publishing.

[103] T. Preston-Werner. Semantic versioning 2.0.0. `https://semver.org`. Accessed: 2022-05-29.

[104] ProductPlan. RICE scoring model. `https://www.productplan.com/glossary/rice-scoring-model/`. Accessed: 2022-05-29.

[105] Project Management Institute. *A guide to the project management body of knowledge*. Sixth edition, 2017.

[106] D. M. Rafi, K. R. K. Moses, K. Petersen, and M. V. Mäntylä. Benefits and limitations of automated software testing: Systematic literature review and practitioner survey. In *7th International Workshop on Automation of Software Test, AST 2012 - Proceedings*, pages 36–42, 2012.

[107] S. Reich and W. Behrendt. Technologien und Trends für Wissensarbeit und Wissensmanagement. *HMD Praxis der Wirtschaftsinformatik*, 44:6–15, 12 2007.

[108] T. Ritchey. *General Morphological Analysis (GMA)*, pages 7–18. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[109] N. B. Ruparelia. The history of version control. *ACM SIGSOFT Software Engineering Notes*, 35:5–9, 1 2010.

[110] A. Saaksvuori and A. Immonen. *Product lifecycle management (third edition)*. Springer Berlin Heidelberg, 2008.

[111] M. Saeki and H. Kaiya. On relationships among models, meta models and ontologies. In *6th OOPSLA Workshop on Domain-Specific Modeling*, 2006.

[112] G. Santoro, D. Vrontis, A. Thrassou, and L. Dezi. The internet of things: Building a knowledge management system for open innovation and knowledge management capacity. *Technological Forecasting and Social Change*, 136:347–354, 11 2018.

[113] Scaled Agile Inc. SAFe lean-agile principles. `https://www.scaledagileframework.com/safe-lean-agile-principles/`. Accessed: 2022-05-29.

[114] R. F. Schmidt. *Software Engineering*. Morgan Kaufmann, 2013.

[115] G. Schuh, H. Rozenfeld, D. Assmus, and E. Zancul. Process oriented framework to support PLM implementation. *Computers in Industry*, 59:210–218, 3 2008.

[116] E.-M. Schön, J. Sedeño, M. Mejías, J. Thomaschewski, and M. J. Escalona. A metamodel for agile requirements engineering. *Journal of Computer and Communications*, 07:1–22, 2019.

[117] Z. Scott and D. Long. One model, many interests, many views. Whitepaper, Vitech Corporation, 2018.

[118] J. Shabi and Y. Reich. Developing an analytical model for planning systems verification, validation and testing processes. *Advanced Engineering Informatics*, 26:429–438, 4 2012.

[119] A. J. Shenhar and D. Dvir. *Reinventing Project Management: The Diamond Approach to Successful Growth and Innovation*. Hardvard Business Review Press, 2007.

[120] A. Silventoinen, H. J. Pels, H. Kärkkäinen, H. Lampela, H. J. Pels, and J. Okkonen. PLM maturity assessment as a tool for PLM implementation process. In *Proceedings of PLM10, The IFIB WG5.1 7th International Conference on Product Lifecycle Management*, 2010.

[121] S. Singh, S. C. Misra, and S. Kumar. Identification and ranking of the risk factors involved in plm implementation. *International Journal of Production Economics*, 222, 4 2020.

[122] Z. Srdjevic, R. Bajcetic, and B. Srdjevic. Identifying the criteria set for multicriteria decision making based on swot/pestle analysis: A case study of reconstructing a water intake structure. *Water Resources Management*, 26:3379–3393, 9 2012.

[123] J. Stark. *Product Lifecycle Management*. Springer London, 2011.

[124] N. P. Suh. Designing-in of quality through axiomatic design. *IEEE Transactions on Reliability*, 44:256–264, 1995.

[125] A. R. Tan, T. C. McAloone, and M. M. Andreasen. What happens to integrated product development models with product/service-system approaches? In *6th Integrated Product Development Workshop*, 10 2006.

[126] L. P. Taylor. *FISMA Compliance Handbook*. Syngress, 2013.

[127] The Open Group. The TOGAF® standard, version 9.2. `https://pubs.opengroup.org/architecture/togaf9-doc/arch/`. Accessed: 2022-05-29.

[128] S. Thomke and E. Hippel. Customers as innovators: A new way to create value. *Harvard Business Review*, 80, 04 2002.

[129] D. Tomar. TQM, ISO 9000, Six Sigma and CMMI project management in business and technology. *Software Engineering*, 2020:1–8, 2020.

[130] D. Tony Liu and X. William Xu. A review of web-based product data management systems. *Computers in Industry*, 44(3):251–262, 2001.

[131] S. Tuck. Is MDM the route to the holy grail? *Journal of Database Marketing & Customer Strategy Management*, 15:218–220, 12 2008.

[132] S. Tyagi, A. Choudhary, X. Cai, and K. Yang. Value stream mapping to reduce the lead-time of a product development process. *International Journal of Production Economics*, 160:202–212, 2 2015.

[133] U.S. Department of Defense. The DoDAF architecture framework version 2.02. `https://dodcio.defense.gov/library/dod-architecture-framework/`, 2010. Accessed: 2022-05-29.

[134] E. Vezzetti, M. G. Violante, and F. Marcolin. A benchmarking framework for product lifecycle management (PLM) maturity models. *International Journal of Advanced Manufacturing Technology*, 71:899–918, 3 2014.

[135] J. Wasson. Configuration management for the 21st century. In *CM conference 2011*, 02 2011.

[136] K. E. Wiegers. *Software Requirements*. Microsoft Press, USA, 1999.

[137] M. Woodbridge. The death of ecm and birth of content services. `https://blogs.gartner.com/michael-woodbridge/the-death-of-ecm-and-birth-of-content-services/`, 2017. Accessed: 2022-05-29.

[138] Z. Y. Wu, X. G. Ming, L. N. He, M. Li, and X. Z. Li. Knowledge integration and sharing for complex product development. *International Journal of Production Research*, 52:6296–6313, 11 2014.

[139] D. C. Wynn and P. J. Clarkson. Process models in design and development. *Research in Engineering Design*, 29:161–202, 4 2018.

[140] Y. Xu, M. K. Malisetty, and M. Round. Configuration management in aerospace industry. In *Procedia CIRP*, volume 11, pages 183–186. Elsevier B.V., 2013.

[141] L. Yang, K. Cormican, and M. Yu. Ontology-based systems engineering: A state-of-the-art review. *Computers in Industry*, 111:148–171, 10 2019.

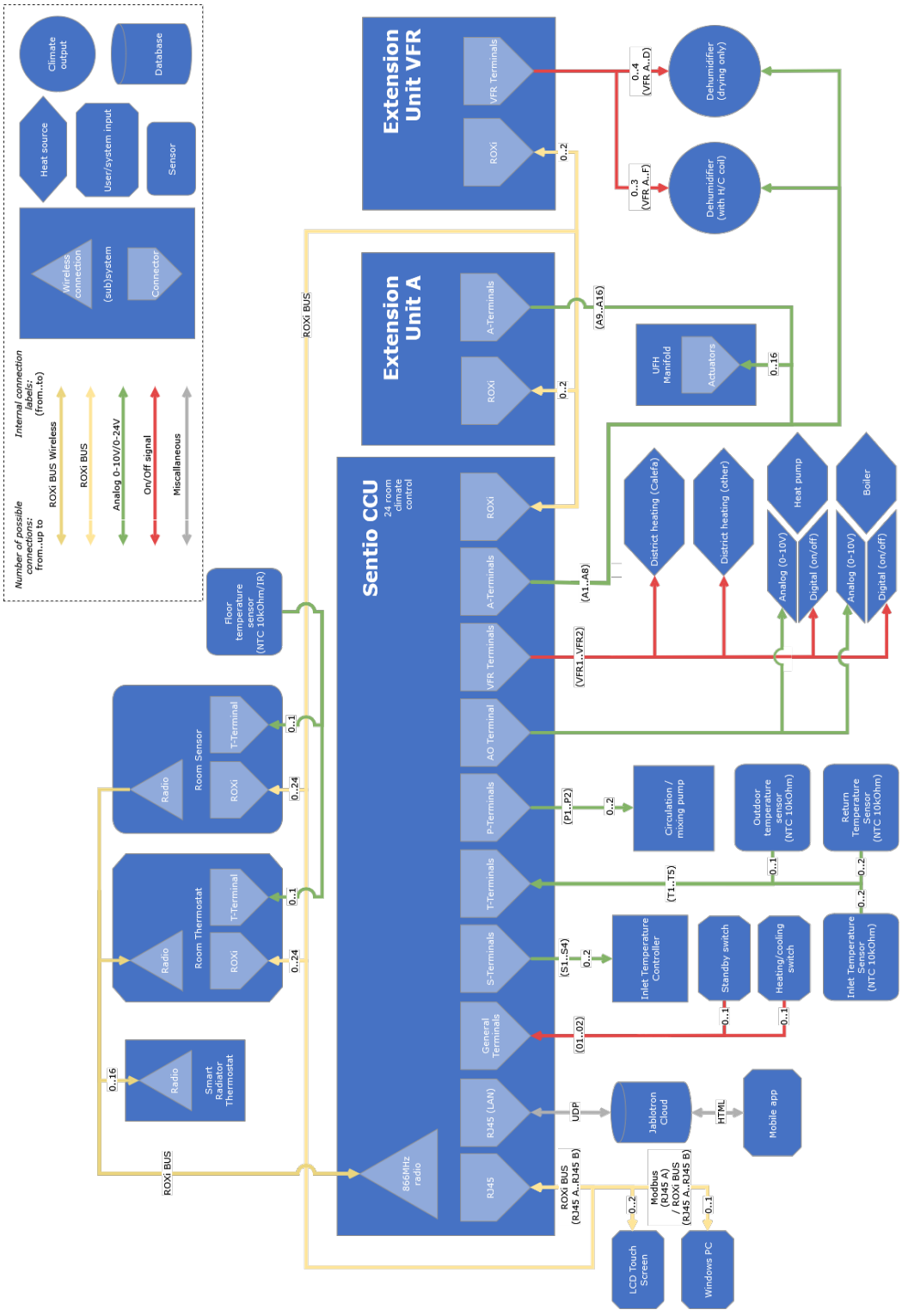
[142] L. Yonglin, Z. Zhi, and L. Qun. An ontological metamodeling framework for semantic simulation model engineering. *Journal of Systems Engineering and Electronics*, 31:527–538, 6 2020.

[143] J. Yoo and Y. Pan. Expanded customer journey map: Interaction mapping framework based on scenario. In C. Stephanidis, editor, *HCI International 2014 - Posters' Extended Abstracts*, pages 550–555, Cham, 2014. Springer International Publishing.

[144] N. N. Zolkifli, A. Ngah, and A. Deraman. Version control system: A review. In *Procedia Computer Science*, volume 135, pages 408–415. Elsevier B.V., 2018.

Appendix A

A technical overview of Sentio is provided on the following page.


Climate output
Database
Heat source
User/system input
Sensor
Wireless connection
(sub)system
Connector
Internal connection labels: (from..to)
ROXi BUS Wireless
ROXi BUS
Analog 0-10V/0-24V
On/Off signal
Miscallaneous
Number of possible connections: from..up to
Extension Unit VFR
VFR Terminals
ROXi
0..2
0..4 (VFR A..D)
0..3 (VFR A..F)
Dehumidifier (drying only)
Dehumidifier (with H/C coil)
Extension Unit A
A-Terminals
ROXi
0..2
(A9..A16)
UFH Manifold
Actuators
0..16
Sentio CCU
24 room climate control
ROXi
A-Terminals
VFR Terminals
AO Terminal
P-Terminals
T-Terminals
S-Terminals
General Terminals
RJ45 (LAN)
RJ45
866MHz radio
(A1..A8)
(VFR1..VFR2)
(P1..P2)
(T1..T5)
(S1..S4)
(01..02)
District heating (Calefa)
District heating (other)
Heat pump
Analog (0-10V)
Digital (on/off)
Boiler
Analog (0-10V)
Digital (on/off)
Circulation / mixing pump
Outdoor temperature sensor (NTC 10kOhm)
Return Temperature Sensor (NTC 10kOhm)
Inlet Temperature Controller
Standby switch
Heating/cooling switch
Inlet Temperature Sensor (NTC 10kOhm)
Floor temperature sensor (NTC 10kOhm/IR)
Radio
Room Sensor
T-Terminal
ROXi
0..1
0..24
Radio
Room Thermostat
T-Terminal
ROXi
0..1
0..24
0..16
Radio
Smart Radiator Thermostat
ROXi BUS
ROXi BUS
UDP
Jablotron Cloud
HTML
Mobile app
ROXi BUS (RJ45 A..RJ45 B)
Modbus (RJ45 A) / ROXi BUS (RJ45 A..RJ45 B)
0..2
0..1
LCD Touch Screen
Windows PC

Appendix B

The comparison of indoor climate control offerings is provided on the following two pages. Below, the price indication scale is included.

Price scale					
Thermostats		UFH Controllers		Radiator thermostats	
<100	€	<250	€	<70	€
100-150	€€	250-350	€€	70-90	€€
>150	€€€	>350	€€€	>90	€€€

Manufacturer	Netatmo (Legrand)	Nest (Google)	Bosch	Sensi (Emerson)	Heatit (Thermofloor)	Tado	Plugwise	Toon (Eneco)
Price	140-200 thermostat, 70-90 radiator	200	100 controller, 70-110 thermostat, 60 radiator	90	110	100 thermostat, 70 radiator	90-150 thermostat, 105-315 UFH	200-275, 4,50 per month
Price category	€€	€€€	€	€	€€	€	€	€€€
Central controller	Thermostat	Thermostat	Controller	Thermostat	Thermostat	Thermostat	Thermostat, boiler controller	Thermostat
App	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Platform compatibility	HomeKit, Alexa, Assistant, IFTTT	Assistant, Alexa, IFTTT	Alexa, Assistant, Homekit, IFTTT	Alexa, Assistant, HomeKit, SmartThings	SmartThings	Assistant, HomeKit, Alexa, IFTTT	Assistant, Homey	Assistant
Zoning	Linking	Linking	TRUE	None	None	Full, limited	Full	None
Product mix size	4	1	5+	1	1	3	5+	1
Connection type	WiFi, BLE	WiFi, BLE	WiFi	WiFi	WiFi, Z-Wave	WiFi, 6LoWPAN with own API	WiFi, ZigBee	WiFi, Z-Wave, own API
HVAC integration	OpenTherm PID	OpenTherm	PI, PWM	None	DIF	OpenTherm, PID	OpenTherm	OpenTherm
Cooling	No	Yes, limited	Underfloor	Yes	No	Yes	Yes	No
Underfloor heating control	Inlet, electric	Inlet, electric	Inlet only	Electric only	Electric only	Inlet only	Full	Inlet, very limited
Humidity control	Measuring only (seperate product)	Full control	Measuring only	Full control	Measuring only	Measuring only	Measuring only	Measuring only
Air quality control	Measuring only (seperate product)	Manual and scheduled	Door/window sensors	Manual and scheduled	No	No	No	Measuring only
Miscellaneous climate control	Heating system self learning, weather and sunlight inclusion	Heating system self learning	Smoke detectors	None	None	Weather inclusion	None	None
Miscellaneous smart features	Velux blinds control, Aldes ventilation control, pattern prediction	Geofencing, pattern prediction	Motion sensors, complete smart home suite including security	None	None	None	None	None
Notes	Renowned design, but limited features for a high price	Popular in the United States. Limited heating compatibility		US, Mainly industrial	Scandinavia		Mostly NL	NL only

Manufacturer	Honeywell	Danfoss	Uponor	Gira	Eve	Ecobee	Rehau	Heatmiser	Wiser (Eberle, SE)
Price	219 controller, 300 UFH, 70 radiator	60 icon thermostat, 140 gateway, 80 ally radiator, 200+ UFH	130 thermostat, 180-320 UFH, 110 radiator	75 thermostat	80 radiator	200 thermostat	110 thermostat, 470+ UFH	70 thermostat, 80 UFH (no actuators)	60-80 thermostat, 60-100 controller, UFH unknown
Price category	€€	€	€€	€	€€	€€€	€€€	€	€
Central controller	UFH Controller	Smartphone/cloud	(UFH) Controller	Thermostat	Smartphone/cloud	Thermostat	UFH Controller	Thermostat	Controller
App	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Platform compatibility	Assistant, Alexa, IFTTT	Assistant, Alexa	Assistant, Alexa	IFTTT	HomeKit	Alexa, Assistant, HomeKit, IFTTT	Alexa	Alexa, Assistant, HomeKit, IFTTT	Alexa, Assistant, IFTTT
Zoning	Full and linking	Full and linking	Full and linking	None	None	Full, limited	Full and linking	Full	Full
Product mix size	5+	5+	5+	5-Jan	2	2	4-Feb	2	4
Connection type	WiFi, Sub-GHz with own API	WiFi, ZigBee, Z-Wave, own API	WiFi	Bluetooth	Thread	Zigbee (module)	WiFi	WiFi	WiFi, Zigbee & MQTT (seperate product)
HVAC integration	OpenTherm	PID	PID, PWM	None	None	None	PID	OpenTherm	OpenTherm
Cooling	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
Underfloor heating control	Full and electric	Full and electric	Full and electric	Full, complex	None	Inlet and electric, very limited	Full	Full	Full
Humidity control	No	Measuring only	Measuring only	No	Measuring only	Full control	Full control	No	Measuring (seperate product)
Air quality control	No	No	No	Measuring only	Measuring only (VOC)	Ventilation control (third party)	No	No	No
Miscellaneous climate control	None	None	None	Blinds control	None	None	None	None	None
Miscellaneous smart features	None	None	None	KNX	None	Pattern prediction	KNX	None	Geofencing
Notes		Gateway		Complete premium smart home solution, very limited climate control		NA focused	No radiator thermostats, UFH only	No radiator thermostats, NO UFH ACTUATORS, mostly UK	UFH in the future. Schneider Electric collab with Danfoss, Somfy, ASSA ABLOY

Appendix C

Non-intrusiveness philosophy

When designing a product with (at first) undefined user interaction, such as an indoor climate control system, it is important to look at the relation the user has with it. This can help shape the interactions to align with the intentions and expectations of the user. In terms of the human-technology relations distinguished by Don Ihde, as described by Peter-Paul Verbeek[1], there is a background relation between the end user and the climate system. This is schematized as *human (technology/world)*. The technology is then part of the world, serving its purpose in the background, like a floor or a fridge. However, when controlling the system, especially through a digital system remotely, the interaction could be described as an alterity relation, where the human 'alters' the state of the world through technology. This is schematized as *human* $\rightarrow$ *technology (world)*. Technology then serves the mediating role between the user and the world; the state of the world is altered using technology.

It is important to align the functions of the indoor climate systems to their respective human-technology relations. The current 'non-intrusiveness' focus for Sentio of Wavin fits the background relation adequately. However, some interaction and user control is required, and minding the alterity relation when doing so can be of added value. If the goal of Sentio is to achieve true non-intrusiveness, the amount of interaction (alterity) is to be minimized, leaving Sentio to provide the optimal indoor climate automatically at all times. Due to technological limitations, this goal is impossible to be achieved instantly. However, the target of minimizing interaction can be pursued at all times, and can translate to design features. After stripping the user interface, leaving only the necessary, desired controls,

[1] Peter-Paul Verbeek. *Beyond interaction: a short introduction to mediation theory*. Interactions 22-3, pp 26-31. June 2015.

the remaining interaction can be streamlined. Cognitive ergonomics play an important role in this user interface optimization problem. By incorporating cognitive ergonomics design guidelines in Sentio development, the mental strain for the user in controlling their indoor climate can be reduced. Additionally, these guidelines can aid in simplifying the installation and/or commissioning process of the system.

The non-intrusiveness approach has implications for the Sentio business model and servitization aspects. A strong marketing and brand presence in users' houses and in the digital ecosystem (e.g. through push notifications) contradicts the non-intrusiveness philosophy. A decision must therefore be made on how to make the user aware of the product and how to keep the customer engaged enough with the brand to decide on purchase without intruding. Servitization can be a solution to this problem. Wavin could offer indoor climate solutions as a service (Climate as a Service, CaaS), with a marketing focus on convenience and efficiency towards the end customer, and a focus mostly on ease of installation towards the installers.

Appendix D

Miscellaneous methods

Domain-driven design

Domain-driven design (DDD) is a method that prescribes context mapping, distillation, and application of large-scale structures to explicitly define the intricacies between a domain model and the implementation of a software design. DDD is often used in software development for clarifying the distinction between a design domain and the design implementation. Furthermore, the context mapping activity of DDD can be aligned with enterprise architecture [79].

Design for X / Design to cost

Design for assembly (DFA), design for manufacturing (DFM), and its combination (DFMA) are methods developed to optimize the manufacturability of products already from the design process. These methods are frequently added as development steps after concept design. Design for assembly focuses on the ease and cost of (dis)assembling the product, whereas design for manufacturing focuses on the material processing costs [45]. Boothroyd and Dewhurst [20] prescribe concrete design practices to optimize assembly and manufacturing. DFMA can be coupled with design to cost, which is a method that applies a cost estimation to design candidates iteratively [45].

Critical path method

The critical path method allows project managers to identify and sequence the activities of a project that are critical to the project schedule. Adhering to this method helps project managers prioritize the most important activities in the schedule [100].

Four actions framework

The four actions of the four actions framework are: reduce, create, raise, and eliminate. Reduce and eliminate aim to avoid conforming to unnecessary industry standards, whereas create and raise focus on factors underappreciated by the industry. It is part of their larger Blue Ocean Strategy, of which the main purpose is to find uncontested market space to render competition irrelevant [76].

Customer-as-innovator

The customer-as-innovator product development approach shifts the customer interface from right after the prototype phase to before the detailed design phase. Incorporating the customer in the innovation process increases the frequency of trial-and-error iterations, speeding up innovation. The customer-as-innovator approach works best when the product is characterized by high customizability, high trial-and-error rates, and the use of rapid prototyping [128].

Value stream mapping

Value stream mapping (VSM) is a process mapping method aimed at resolving root causes of bottlenecks in sequential processes. VSM was originally developed for manufacturing process but is adaptable to the generally less structured product development process [139]. VSM works by mapping the current process state and identifying bottlenecks, transforming the map into a desired future state. This has been successfully applied to product development, reducing the product development lead time [132]. Like the value stream map itself, the Takt time principle of lean production can be applied to product development by parsing them from the VSM [96]. Here, the similarity with Scrum sprints is noticeable.

Dynamic systems development method

The dynamic systems development method (DSDM) is an early agile development method. DSDM is a combination of several lightweight tools and practices such as incremental development, the MoSCoW (must, could, should, won't have) technique, modelling, and prototyping. Stemming from rapid application development, it is a software engineering method [102]. Dynamic product development (DPD) can be seen as its physical product development counterpart, being an iteration-driven approach focusing on allowing a concept to be adjusted continuously throughout a project [139]. Both methods emphasize the necessity of strong leadership.

Miscellaneous models

Project management triangle

The project management triangle (also referred to as the iron triangle) models the three main constraints of a project: quality, cost, and time [9]. This model highlights the mutual exclusivity of these constraints, meaning that improving on one criterion always comes at the cost of one or both of the other criteria.

NTCP model

The NTCP model structures the innovative characteristics of a project in four dimensions: novelty, technology, complexity, and pace. Novelty is divided into derivative, platform, and breakthrough categories. Technology is a measure of technological uncertainty. Complexity indicates general project risk due to assembly, system, or array (system in context) intricacy. Finally, pace indicates the time sensitivity and urgency of the project [119]. Overall, this model can visually characterize the potential benefits and risks of projects for comparison.

Miscellaneous tools

N2 diagram

The N2 (or N^2, or N-squared) diagram is a tool used to identify and note interfaces within a system. The N2 diagram is a square table with along the diagonal axis the primary functions or subsystems of the system. In clockwise fashion, crossing rows and columns denote interfaces between the subsystems [19]. Below, in table D.1, this is demonstrated.

Table D.1: N2 diagram example

F1	Interface from F1 to F2		
	F2	Interface from F2 to F3	
		F3	
Interface from F4 to F1			F4

Nine window diagram

The nine window diagram is a TRIZ tool. It is used to understand the system, its subsystem and its context in a past, present, and future state [63]. The use of a nine window diagram facilitates the discussion of the future of a system as a whole and its consequences on and

influences by the subsystems and context [19]. An example of a nine window diagram is provided in figure D.1.

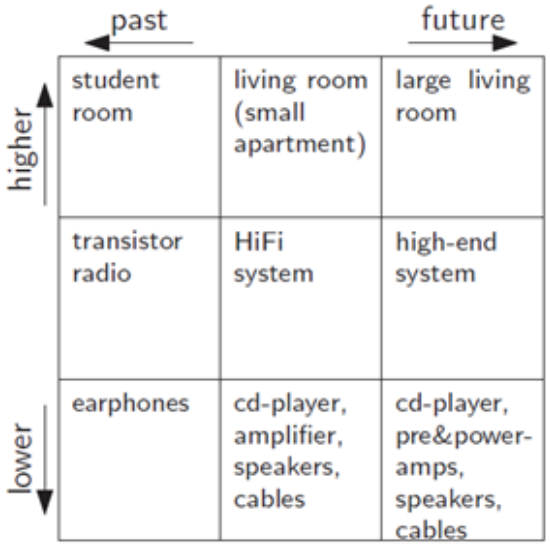


Figure D.1: Nine window diagram [19]

Decision tree analysis

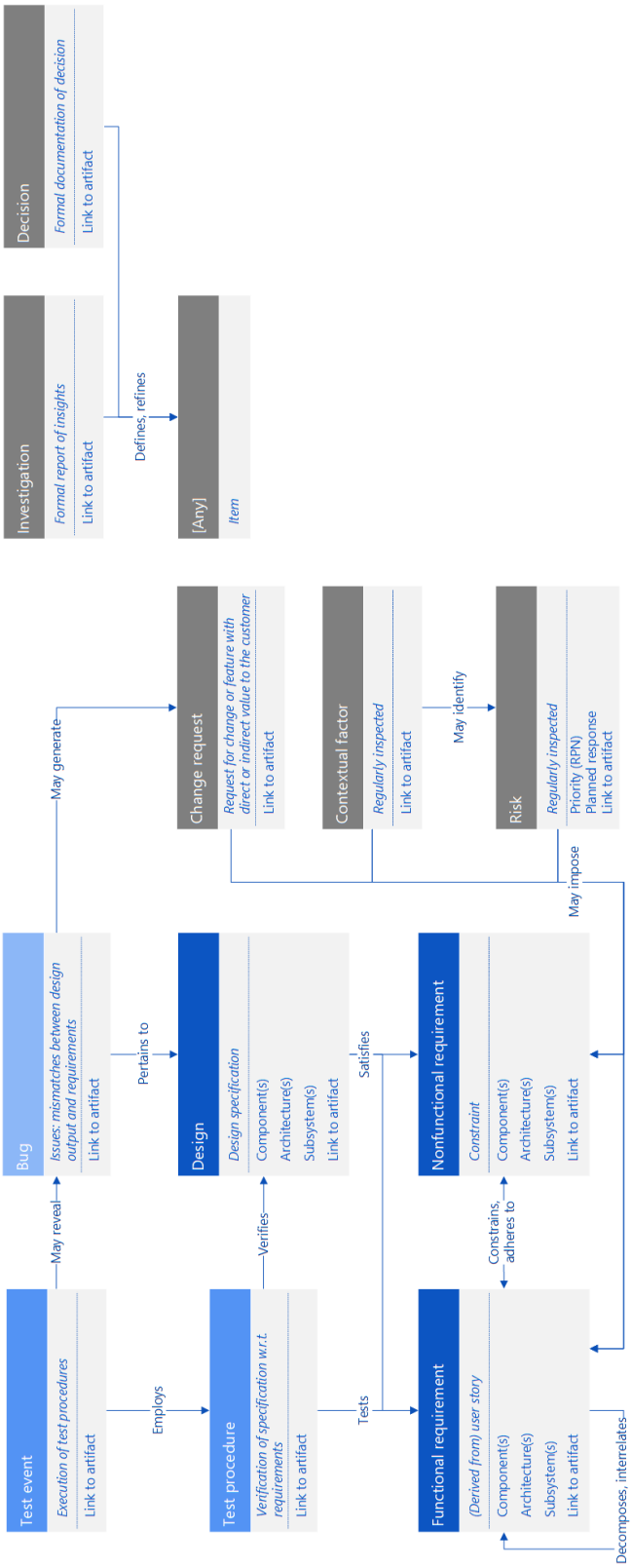
In a decision tree analysis, scenarios are systematically developed based on potential decisions. These scenarios can be of qualitative nature to plan for scenarios, but numeric values can also be added to branches to quantifiably assess various scenarios [19].

Customer journey map

The customer journey map is a tool that illustrates the scenario of a customer experiencing the complete service of the organization. In a journey map, the interactions between the customer and the organization are highlighted [143]. This tool can aid in systematically identifying the value delivery points of a service.

Appendix E

On the following page, the system development metamodel is shown.


Decision
Formal documentation of decision
Link to artifact
Investigation
Formal report of insights
Link to artifact
Defines, refines
[Any]
Item
Change request
Request for change or feature with direct or indirect value to the customer
Link to artifact
Contextual factor
Regularly inspected
Link to artifact
May identify
Risk
Regularly inspected
Priority (RPN)
Planned response
Link to artifact
May generate
May impose
Bug
Issues: mismatches between design output and requirements
Link to artifact
Pertains to
Design
Design specification
Component(s)
Architecture(s)
Subsystem(s)
Link to artifact
Satisfies
Nonfunctional requirement
Constraint
Component(s)
Architecture(s)
Subsystem(s)
Link to artifact
May reveal
Verifies
Constrains, adheres to
Test event
Execution of test procedures
Link to artifact
Employs
Test procedure
Verification of specification w.r.t. requirements
Link to artifact
Tests
Functional requirement
(Derived from) user story
Component(s)
Architecture(s)
Subsystem(s)
Link to artifact
Decomposes, interrelates