Incorporating Model-Based User Requirements in Product Development

by

Luke Niewiadomski

A thesis submitted in partial fulfilment
of the requirements for the degree of
Master of Science
(Industrial and Systems Engineering)
in the University of Michigan-Dearborn
2021

Luke Niewiadomski

lniewiad@umich.edu

ORCID iD: 0000-0002-6890-7341

Dedication

This work is dedicated to my father, Mitch Niewiadomski, who greatly inspired my interest in engineering and provided the foundation for my abilities in the field (and so much more).

Acknowledgements

A deep gratitude to those who contributed to this research, particularly Dr. Maxim for his brilliant and enthusiastic guidance together with his steadfast effort. A similar thanks to Dr. Bao as a powerful source of inspiration, insights, and sincere support. An additional sincere thanks to Dr. Feng Zhou for his astute and helpful feedback on the work.

A great number of people influenced this work both directly and indirectly, most notably the dozens of incredible work colleagues adjacently involved in the effort. They are thought of often with meaningful appreciation.

Table of Contents

List of Figures

List of Tables

List of Terms

The following definitions and abbreviations are applied in this thesis.

Definitions

Acceptance Testing

A test level that focuses on determining whether to accept the system (ISTQB Glossary).

Agile Development

A software development approach based on iterative development, frequent inspection and adaptation, and incremental deliveries, in which requirements and solutions evolve through collaboration in cross-functional teams and through continual stakeholder feedback (ISO 26515).

Case Organization

The anonymous company at which the research was performed in this thesis.

Context of Use

The combination of users, goals and tasks, resources, and environment (ISO 9241).

Derived Requirement

A requirement deduced or inferred from the collection and organization of requirements into a particular system configuration and solution (ISO 29148).

Effectiveness

Accuracy and completeness with which users achieve specified goals (ISO 9241).

Efficiency

Resources used in relation to the results achieved (ISO 9241).

Evaluation

The critical assessment, in as objective a manner as possible, of the degree to which a service or its component parts fulfills stated goals (St Leger and Wordsworth-Bell).

Model-Based Systems Engineering

Formalized application of modeling to support system requirements, design, analysis, verification, and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases (INCOSE-TP-2004-004-02).

Requirement

A statement that translates or expresses a need and its associated constraints and conditions (ISO 29148).

Requirements Validation

Confirmation that requirements (individually and as a set) define the right system as intended by the stakeholders (EIA632).

Requirements Verification

Confirmation by examination that requirements (individually and as a set) are well-formed (ISO 29148).

Satisfaction

The extent to which the user's physical, cognitive and emotional responses that result from the use of a system, product, or service meet the user’s needs and expectations (ISO 9241).

Traceable

Having components whose origin can be determined (ISO 15289).

Usability

The extent to which a system, product or service can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use (ISO 9241).

User Requirement

The requirements for use that provide the basis for design and evaluation of interactive systems to meet identified user needs (ISO 25065).

User

A person who interacts with a system, product, or service (ISO 9241).

Validation

Confirmation, through the provision of objective evidence, that the requirements for a specific intended use or application have been fulfilled. Note, validation is the set of activities ensuring and gaining confidence that a system is able to accomplish its intended use, goals and objectives (e.g. meet stakeholder requirements) in the intended operational environment. (ISO 9000)

Verification

Confirmation, through the provision of objective evidence, that specified requirements have been fulfilled. Note, verification is a set of activities that compares a system or system element against the required characteristics. (ISO 9000)

Abbreviations

HCD	Human-Centered Design
HTA	Hierarchical Task Analysis
IQR	Interquartile Range
INCOSE	International Council on Systems Engineering
IRB	Institutional Review Board
ISO	International Standards Organization
K-W	Kruskal-Wallis
MBSE	Model-Based Systems Engineering
MOE	Measure of Effectiveness
NIST	National Institute of Standards and Technology
NFR	Non-Functional Requirement
RADC	Rome Air Development Center
RE	Requirements Engineering
SEBoK	Systems Engineering Book of Knowledge
SF	System Function
SR	System Requirement
SysML	Systems Modeling Language

UR	User Requirement
VA	Validation Attribute
V&V	Verification & Validation

Abstract

System complexities continue to expand in the unfolding technology revolution, as does the array of user needs, interactions and quality concerns. Human-centered aspects are increasingly missed during product development, leading to poorly designed products and rework. Modern requirement methods are evolving in response to these trends and to improve the user experience of new products.

This thesis proposes a method for the early incorporation of User Requirements (URs) into the engineering development process. It attempts to synergistically harness the benefits of human-centered design, requirements engineering, and modeling. The framework leverages work from ISO standards, along with a focus on pragmatic application and best practices in model-based systems engineering. Its ultimate intention is to provide an effective means to capture and meet user needs during the development of an interactive system.

Research on the proposed method was performed within a major automotive company. Towards understanding the impressions of URs in practice, a survey was distributed to 59 engineers at the organization on their impressions of the proposed URs approach. In parallel, URs were incorporated into three advanced technology projects at during a mid-cycle design iteration, followed by a study of the subsequent changes made to the system requirements and test plans in response to the inclusion of URs.

The survey results revealed the UR proposal was well received, with an overall 96% reporting favorable expectations for a system development project. The case studies supported these results by revealing a direct impact to design functionalities, quality aspects, and validation activities. System requirements increased by 15%, and validation attributes by 38%.

Chapter 1. Introduction

The orientation of this work is to improve the incorporation of human-centered design methods in the development of interactive systems, with the intention to better meet user needs, and to help manage the complexity of the information involved in such an endeavor. Modern engineering practices are too difficult to sufficiently navigate without aids and frameworks to guide development. The work presented here attempts to offer an important piece to fit within this overall venture, on the capturing of User Requirements (URs). These methods aim to improve both the quality and functionality of a system early in the development process.

In this research, a proposed framework and process workflow are put forward as an applicable method for the incorporation of URs into a product development lifecycle. The URs are developed based on a variety of upstream human-centered content, and in turn become the basis of the system requirements and validation activities. A model-based systems engineering (MBSE) approach is leveraged to organize URs as discrete and traceable requirements objects.

This work contributes to the engineering domain in the following ways:

- By providing a model-based UR framework
- By denoting a viable process for UR incorporation into an overall workflow
- By providing evidence on the value of URs in a project
- By offering a critique of benefits and limitations regarding URs

The hope is that the knowledge presented here provides some practical usefulness to real-world applications, fostering a strengthened human-centered approach for the design and development of new products.

1.1 Problems and Opportunities

Developing high-quality products benefit from a clear understanding of the user needs, wants, and expectations. These human-centered aspects are typically amorphous and multifaceted, making them challenging to capture. Consequently, this type of information is often described in a free-form manner without sufficient clarity. In other cases, interactive systems are defined by the developers themselves without much external input [1].

Even when usability and user experience content is gathered before a system is designed, the knowledge is rarely captured in the form of requirements [2]. This is largely due to the many difficulties involved in defining URs and integrating them into a system development lifecycle. Users generally have only a vague idea of what they want, and only when they receive a finished product does it become clear what to turn into a requirement [3]. URs attempt to provide a degree of foresight to displace this hindsight, but the methods to realize this goal are neither obvious nor trivial.

The importance of requirements has been well documented. Many if not most of software failures can be attributed to insufficient requirements [4]. A long-term study by the Standish Group revealed that development projects fail in large part from to a lack of user involvement and incomplete requirements [5]. In other instances, even if a system is defect-free, extra functionalities may be provided that are unfitting or unwanted by a user. For example, one study of banking systems uncovered that 70% of available functions remained completely unused [6].

Therefore, efforts to improve upon requirements methods can prove extremely worthwhile. While requirements development only accounts for 10–15 percent of the overall system development cost, incurred costs for later changes increase disproportionately across the lifecycle [3], as seen in Figure 1. Efforts to discover issues and opportunities early in the process (such as with URs) are well deserved from an economic perspective.

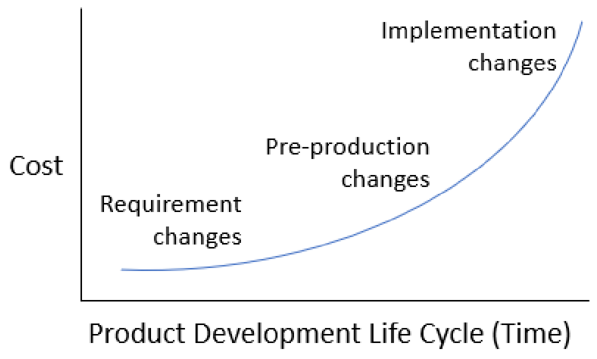


Figure 1
Costs for Product Changes across Development Lifecycle

As an additional pitfall, requirements may be clearly captured yet provide little influence on the system development process. URs often do not link to the design requirements and design aspects [7] despite the effort expended to create them. Consequently, system requirements and design decisions are often made with a disconnect to the driving factors behind the system [8].

Adequately incorporating requirements into the development workflow can also be problematic. Complex sets of information are inherently challenging to communicate and manage, and poor engineering decisions become likely without proper requirements handling. Requirements traceability is therefore critical in order to ensure the system meets expectations and to manage changes [9]. URs must also tightly connect to the user needs within the operating context [10]. Without the proper interconnection of information, communication subsequently involves a large degree of translation and decomposition, both of which are highly susceptible to confusion and mistakes. Traceable model-based requirements are frequently employed as a mitigation tactic for these issues.

Research has also supported the importance of validation efforts during the system requirements process in order to improve quality and produce changes earlier in the development lifecycle [11]. However, test cases for complex systems typically need to be driven by well-defined artifacts such as requirements. Verification testing is commonly based upon requirements, and user validation testing would expectedly benefit from following a similar requirements-driven approach.

Growing trends in business and technology further exacerbate these issues and give stronger credence to a UR mindset [12]. Systems are becoming increasing complex, as are their associated user-related aspects. Fast-changing technologies continue to emerge, with strong competition in industry to produce products that amply meet user needs and produce delight. Methods to better manage UR content seem highly suitable to the modern engineering domain and are becoming more critical over time.

1.2 Purpose of the Work

The aim of this work is to improve product design by improving the methods used to specify them. In this thesis work, a human-centered approach is presented to capture and integrate URs into an early product development effort. This serves as an attempt to address the problems mentioned previously: challenges in generating URs, system designs without traceability to user needs, weakly defined validation efforts, and outdated information management practices.

Ultimately, the method presented in this research aims to simultaneously harness the benefits from the fields of Requirements Engineering (RE), Model-Based Systems Engineering (MBSE), and Human-Centered Design (HCD) through a UR-centric perspective. Several existing approaches attempt to provide similar outcomes [13], but none have been identified which bring these fields into a UR-centric framework and workflow.

As a derived purpose, this research assesses the proposed method on its effectiveness and suitability in industrial applications. A focus on pragmatic value is taken as fundamental for the context for this work. Existing research, ISO standards, and case-studied practices are accordingly adapted into a readily applicable method which does not rest on theory alone.

Finally, another goal of this work is to enhance the experience of the engineers and the organizations involved. Communication of user-related information is characteristically challenging and often performed with degrees of tension and uncertainty. The availability of a clear and applicable method may alleviate these issues to some degree and support a unified effort across a wide array of contributors: users, user experience professionals, human factors engineers, systems engineers, analysts, developers, testers, project managers, business stakeholders, and more.

To summarize the above commentary, the overall purpose of this work is organized into a set of primary objectives:

- To identify a viable approach to capture traceable URs in early product development
- To propose an adaptable process to embed URs into a project workflow
- To evaluate the UR proposal for workload, understandability, and expectations from involved participants
- Investigate the direct impact of URs through case studies on actual project work

1.3 Potential Benefits of URs

In the sections below, several potential benefits of URs are discussed. A case is made for using URs, particularly through an early incorporation into the design process and by making them an integral part of the traceability effort. Additionally, URs can provide a substantial impact on validation testing through their proper incorporation.

1.3.1 General Benefits

According to National Institute of Standards and Technology (NIST), the following benefits from URs have been put forward [14]:

- Increased likelihood of project success
- Reduction in development time, effort, and cost
- Creating a baseline for validation and acceptance criteria
- Providing a basis for product improvements
- Better tracking and organization of requirements

The following additions also become apparent based on UR contributions within a HCD approach [3]:

- Improved selection of system functionalities
- Improved project quality

1.3.2 Benefits to a Project

To elaborate on the project benefits, early-stage capturing of URs can lead to a substantial cost and timing impact on the development of a system. Estimates include a 1000-fold cost reduction for discovery of issues during concept development as compared to during product testing and production [15]. Much technical debt is alleviated early on and project success becomes more likely.

Requirements also serve as a suitable method for information management. The organization of specification content is benefited, as is maintainability and communication between project teams. These RE advantages can be realized by the HCD community through the incorporation of URs.

1.3.3 Benefits to System Design

Product design benefits of URs are expected through a stronger human-centered approach. These include improvements to product quality and more suitable system functionalities. Essentially, URs are able to provide the basis for a system design [16] and improved quality in meeting customer needs [17]. These improvements consequently lead to a better product and a reduction in rework.

The conviction is that human-centered content must be well-captured early and throughout a development process, with sufficient quality and completeness to effectively influence the product design. Without clear URs, system design aspects could easily be missing, wrong, unclear, irrelevant, or insufficient.

1.3.4 Benefits to Validation Testing

URs also provide the basis for the evaluation of a system from the user's perspective [16], through validation and acceptance testing. This type of testing is challenging to realize without proper input information (such as URs). In comparison, verification testing is classically requirements-driven, and thus high-quality requirements directly lead to pertinent testing activities. When URs are created, validation could be similarly performed as an evaluation of requirement conformance, ensuring test coverage across a comprehensive set of URs. Subsequently, less distinction is needed between the terms verification and validation if approached in this manner.

1.4 Research Questions

This work is separated into three efforts, each with corresponding research questions, listed below. These are expanded upon in Chapter 4 (Research Methodology) and addressed in Chapter 5 (Research Results).

- <u>Developing a UR Framework and Process</u>
    - What framework and process workflow would be suitable for model-based URs?
- <u>Evaluating UR Expectations (Survey Study)</u>
    - Would URs benefit a system development effort?
    - Would URs benefit particular aspects of a system?
    - Do URs only benefit complex interactive systems?
    - Are certain types of URs more valuable than others?
    - Is the UR framework easy to incorporate into a workflow?
    - Are UR traceability relationships important?
    - Which UR traceability relationships are important?

- Evaluating UR Impact (Project Study)
    - Does the incorporation of URs lead to changes in the system design?
    - Does the incorporation of URs lead to changes in validation testing?
    - Does the incorporation of URs lead to unmanageability of requirements data in a project?

1.5 Research Design

The UR proposal was evaluated through two avenues at a major company, by means of a survey and project case studies. An overview is below, with details covered in Chapter 4.

The first effort involved a 34-question survey (see Appendix) to gather their impressions and expectations of the UR method. The research effort involved an online distribution of the survey to engineers and analysts at the organization, followed by statistical and qualitative analysis of the results.

The second effort involved three cases studies, in which the UR framework was implemented into active design projects at the organization, with an analysis on the downstream impact delivered by the URs. The research method utilized quantitative measurement of the UR-driven influence to system requirements and to validation test plan attributes.

The research occurred within the workplace of the investigator (a major automotive company), with the company name and project details kept confidential for proprietary reasons. This company is referred to as the Case Organization within this document. Involvement was focused in the field of interactive automated driving technologies. Survey and project participation spanned the departments of research, product development, and user experience.

1.5.1 Scope of the Work

The boundary of the study focused on URs within a system development workstream, confined to an MBSE approach for complex interactive systems. The effort targeted advanced technology work at a single Case Organization.

It is also helpful to specify what this work is not considered to be. This work does not provide methods for user research, user experience efforts, or HMI design. Nor does it provide system design and development guidance. It also does not specify a framework or model for validation testing. Regarding URs, it does not serve as a traceability metamodel or other specific MBSE profile. In short, it constrains itself to the URs themselves and does not stretch to cover the surrounding related information.

The scope is also contextualized as providing a reference method, as opposed to a strict prescriptive formula. The UR framework and process are designed as guidance aids, meant for practical adaptation into industry applications. This can be viewed in contrast to systematic processes or automatable procedures, which it is not intended to represent.

1.5.2 Role of the Researcher

The researcher was the primary source of the UR framework and process, with support from university advisors and colleagues in the Case Organization. He also led the planning and conducting of the corresponding studies to evaluate the UR proposal.

The problem statement and UR proposal was generated out of his direct experiences at the Case Organization and are thus tailored accordingly for implementation at that specific company. The findings are therefore not necessarily of direct pertinence to another organization or industries in general.

1.6 Thesis Organization

The remaining chapters are organized as follows:

- Chapter 2: Literature Review. A collection of published work is summarized on the topics of relevance to this research, including user-related content, requirements, quality models, and model-based systems engineering.
- Chapter 3: User Requirements Proposal. An overview is given of the proposed UR framework, along with a suitable and adaptable process for its incorporation into an engineering workflow.
- Chapter 4: Research Methodology. An evaluation of the proposed framework and process is performed using a survey and project case studies at the Case Organization.
- Chapter 5: Research Results. Results from the survey data and project studies are presented and analyzed, with connection to the research questions.
- Chapter 6: Discussion. Commentary on the findings is provided, including apparent indicators for the benefit of the UR framework, interpretations on its application in practice, lessons learned, and threats to the validity of the findings.

- Chapter 7: Conclusion. A brief summary of the research implications is presented, along with additional applications and recommendations for further research.

Chapter 2. Literature Review

This thesis is built upon a synthesis of several complimentary research topics, each with multifaceted aspects. These are organized in the sections to follow, covering the following topic areas:

- User-related informational content
- Requirement types and aspects
- Engineering processes
- Model-based systems engineering

2.1 User-Related Informational Content

Before discussing URs or requirements in general, it is helpful to cover existing methods for capturing information about users and their interactions with systems. This information may be used in practice as (a) the basis for developing URs, (b) in place of URs, or (c) complimentary to URs.

User-related content is available in a vast array of content types and formats and must be handled in a manner specific to the workflow in which it is captured and communicated. The sections below cover a few commonly used types of information relevant to the creation of URs, yet these are certainly not comprehensive or necessarily best practices.

2.1.1 User Research

Many avenues are available for the gathering of user information and identifying user needs. Passive research, online studies, or face-to-face means can be employed, typically in combination. Specifically, these can include user clinics, interviews, surveys, user needs analysis, market appraisals, group task analysis, technical experiments, existing research reviews, field studies, and focus groups [18, 19]. The outputs of these efforts typically lead to downstream methods to analyze and organize the findings (such as those mentioned in the sections below), depending on the type of activity being performed and its context.

2.1.2 User Stories

User stories are a popular technique within Agile development to capture a particular functionality from the user's perspective in natural language [20]. A set of various stories can be used to descriptively capture the user-centered aspects of a system in use. Captured in a sentence, each story usually includes the user's role, benefit received, and capability employed. Compared to URs, this approach lacks testable criteria, and focuses on functional behaviors without capturing system quality aspects sufficiently. As such, one or more URs could be derived from a user story.

2.1.3 Storyboards

Sometimes confused with user stories, a storyboard provides a sequential depiction of a usage scenario, combining both graphics and text. It reveals how a system is used in its operating context, with illustrations that convey information that is often difficult to express with text [1]. This makes it a powerful communication tool and a valuable source of input material for the creation of URs.

2.1.4 User Flows

User flows are a specialized form of flowchart to depict the user interaction with the system over time. It can expand upon the sequential nature of single scenario depictions (such as storyboards) by providing decision nodes to reveal multiple possible paths. These can expose the user's tasks and choices; however, other methods may be needed to uncover these items if they are not already captured.

2.1.5 Hierarchical Task Analysis

A commonly adopted method to determine the user's tasks is the Hierarchical Task Analysis (HTA) [21], which breaks down the user goals into sub-goals until the desired level of detail is achieved. The HTA exposes both general and specific user needs within the context of using the system and is therefore well suited to generating functional requirements. Using this goal-oriented methodology helps with understanding the user's role and their actions, as well as the undesirable outcomes that may occur when a goal is not met.

2.1.6 Acceptance Criteria

To determine if a system meets human-centered aspects (both functional and quality), acceptance criteria are commonly used. These capture the evaluation attributes from the stakeholder perspective (often the user/customer) and serve as the targets for acceptance testing. This form of testing validates if a black-box system is built correctly and completely, and with sufficient quality.

2.1.7 Measure of Effectiveness

Another quality metric is the Measure of Effectiveness (MOEs) used commonly within the U.S. Department of Defense. These are defined as measurable attributes which related to performance and directly correspond to the desired outcomes of a system [22]. Thus, MOEs are akin to acceptance criteria for user-centered outcomes. These measures can be broken down into sub-categories of Measures of Performance and Measures of Suitability. However, MOEs do not cover the constraints or functional capabilities of a system.

2.2 Requirement Types and Aspects

Requirements can come in many forms, be organized in several categories, and incorporate numerous dimensions. These topics are overviewed in the sections below, covering functional and quality requirement types, and details on URs themselves as a form of stakeholder requirement. Finally, an overview of quality categories is given, covering various quality models.

2.2.1 Functional Requirements

Functional requirements capture the "necessary task, action or activity that must be accomplished" [23], or otherwise described as what the system "must be able to do or perform" [24]. These define the use cases, system functionalities, and the results required. They are generally used to describe what and how a system performs, as opposed to quality-related aspects which cover how well an activity is performed.

From a human-centered perspective, a functional-UR would describe a need for a particular function to be performed or achieved in service of the user, without qualitative details. The verification process is typically straightforward (i.e. does the functionality exist or not).

2.2.2 Non-Functional (Quality) Requirements

As compared to functional requirements, non-functional requirements (NFRs) cover various quality aspects of a system. There is not a universally agreed definition for NFRs [25], however a common definition is a requirement that "pertains to a quality concern that is not covered by functional requirements" [26]. Understanding quality-related requirements requires a broader discussion as compared to functional requirements. Quality aspects are commonly more challenging to categorize and to determine what is best to capture in the form of a requirement.

The practical importance of NFRs has been readily acknowledged by industry in relation to product development [27]. Lack of NFRs, or lack of integration with functional requirements, often leads to project delays, substantial costs, and challenging rework [28, 29].

Testing non-functional requirements can be less clear. For instance, soft goals (an objective without clear-cut criteria) are often not definitively met but are rather satisficed [30] based on a comparison of evidence in support or against whether the goal is met. This highlights a need for a different approach in validating soft goals or NFRs, as compared to hard goals or functional requirements.

Some researchers and practitioners note that the term "non-functional requirements" is poorly defined and disparaging, and prefer the use of the term "quality requirements" instead [16]. This language choice is agreeably adopted for this thesis.

2.2.3 Stakeholder Requirements

Besides organizing requirements into functional vs. quality, requirements can also be categorized based on their level of abstraction (e.g., high-level, logical, detailed) or their source type (e.g., business, user, technical). This section discusses how a UR can be perceived as a form of high-level stakeholder requirement type.

Regarding high-level requirements, many types are found in practice. These include business requirements, regulatory requirements, architectural requirements, and finally, relevant to this work, stakeholder requirements [31]. URs are commonly considered as a specialization of stakeholder requirements [32], as the user is a direct stakeholder for the system (note, other stakeholders include: supporters, developers, producers, trainers, maintainers, disposers, acquirers, customers, operators, supplier organizations and regulatory bodies [33]). An initial understanding of URs can come from a study on existing literature classified under the stakeholder requirements category.

Per the Systems Engineering Body of Knowledge (SEBoK) from INCOSE, stakeholder requirements are performed after the business or mission analysis, before the formation of system requirements, and with traceability to support iteration and consistency [24]. They serve the following roles:

- Form the basis of system requirements
- Form the basis for validation and acceptance
- Become a reference for integration and verification
- Support communication between groups (e.g., between the stakeholders and designers)

SEBoK also provides tips on proven practices for capturing these requirements:

- Involve users in the process of creating stakeholder requirements
- Provide rationale for each stakeholder requirement
- Analyze the stakeholder requirements before creating system requirements
- Use modeling techniques and requirements management tools

Additional research on stakeholder requirements uncovers similar content, however there is limited literature on how to distinguish URs or how to treat them separately from other stakeholder requirements. This topic is addressed in the following section.

2.2.4 User Requirements

URs are considered a subset of an overall stakeholder requirements set, tailored to capture user needs, wants, and expectations. Non-user stakeholder concerns are intentionally omitted, such as certain business needs (e.g., cost constraints), development and testing needs (e.g., deployment risks), external goals (e.g., environmental concerns), and more. Also, URs are differentiated from system requirements in that URs represent the user's perspective without specifying a technical solution.

2.2.4.1 ISO 25065

The primary source of research for URs in this work is the recently published ISO 25065 standard on the user requirements specification [34]. Little guidance is otherwise published regarding the specification and syntax of URs themselves [16].

ISO defines URs as a "set of requirements for use that provide the basis for design and evaluation of interactive systems to meet identified user needs". They are "derived from user needs and capabilities in order to allow the user to make use of the system in an effective, efficient, safe and satisfying manner", and thus include both functional and non-functional aspects. URs describe the desired outcomes of a system from the user's perspective, without prescribing a particular system solution.

Two types of URs are defined by ISO:

1. **User-system interaction requirements**, expressing what interactions the user is able to perform towards the intended outcomes of use
2. **Use-related quality requirements**, expressing the effectiveness, efficiency, satisfaction, or other quality-related concerns

This breakdown aligns well with the separation between functional and quality requirements described previously. Clear-cut needs are captured towards defined outcomes (functional) in conjunction with criteria around the performance of meeting these needs (quality).

ISO 25065 also provides a workable syntax for writing URs, using the user as the perspective subject (i.e. "The user shall…"). This clearly differentiates from system requirements which take the system as the subject of the test requirement sentence (i.e. "The system shall…").

Each UR is to fit into the structure of "With the [system], the [user or user group] shall be able to (or shall be satisfied with) [outcome] [with criteria] [under conditions]". Functional requirements are recommended to include (a) the user or user group as the subject, (b) the applicable goal or task, (c) the intended outcome (using terms such as "to recognize", "to input", "to select", "to receive"), and (d) applicable conditions. Quality requirements modify part (c) to capture the effectiveness/efficiency/satisfaction of the outcome along with criteria.

For an ATM machine, the following examples conform to the above guidance:

- Functional-UR: "With the ATM, the user shall be able to transfer funds between accounts."
- Quality-UR: "95% of users shall be satisfied with the transfer interface."

URs also form the basis for the user-system interaction and the acceptance criteria [16]. The illustration in Figure 2 summarizes the overall incorporation of URs amongst the other user-centered artifacts, adapted from the first draft of ISO 25065.

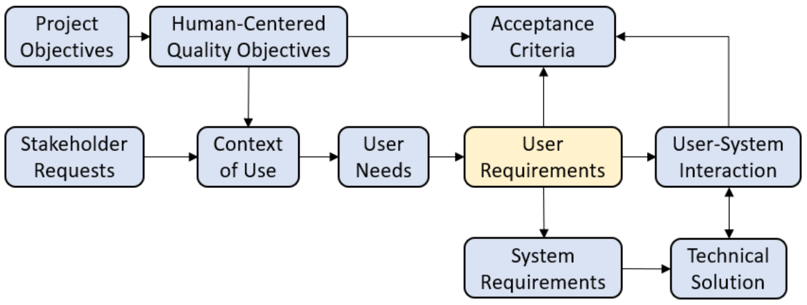


Figure 2
User Requirements Information Flow from ISO 25065 draft

2.2.4.2 NISTIR 7432

In addition to the ISO material outlined above, NIST provides a publicly available internal report, NISTIR 7432: Common Industry Specification for Usability--Requirements. The scope is strictly focused on usability, omitting functional aspects and other quality concerns. Compared to ISO 25065, usability is maintained as a parent category for effectiveness, efficiency, and satisfaction.

NIST distinguishes quality requirements on usability into three distinct levels, corresponding to increasing precision and completeness in the requirement for the performance and satisfaction criteria. Also, consideration is made for the test method under which the usability requirement will be evaluated, which also carries increasing detail with each level. See Table 1 below.

Level	Usability Requirement	Test Method
Level 1	Includes usability criteria with relative importance	Identified test method types
Level 2	Criteria include target values	Preliminary test method for criteria
Level 3	Criteria include specific values	Complete test protocol

Table 1
Levels of Usability Requirements from NIST

2.2.5 Quality Models

Creating URs to cover quality aspects can be challenging, as it requires consideration for a vast number of interrelated aspects. A "quality model" provides a framework for understanding these aspects, by organizing various categories of system quality into useful groupings. Various models exist in practice, with several examples shown below. There is not a one-size-fits-all quality model, but rather multiple perspectives based on pragmatic needs and research evolution.

2.2.5.1 Historical Models

Early efforts to decompose the aspects of software quality can be found from the 1970s, captured as the McCall Quality Model [35], adapted in Figure 3. The attempt was made to sufficiently breakdown the overall quality concerns into testable pieces, which together combine to cover the full gambit of software quality. Three human-centered aspects serve as the foundation: product revision, product transition, and product operation.

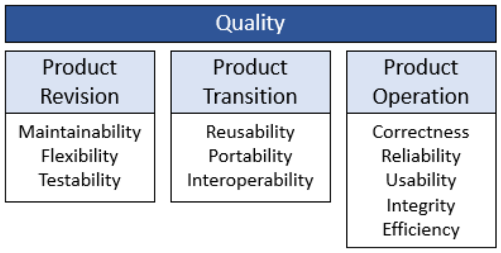


Figure 3
Hierarchy of Software Quality from McCall

Boehm's Quality Model extends the McCall model by incorporating additional categories such as understandability, validity, and more [36]. See Figure 4 below.

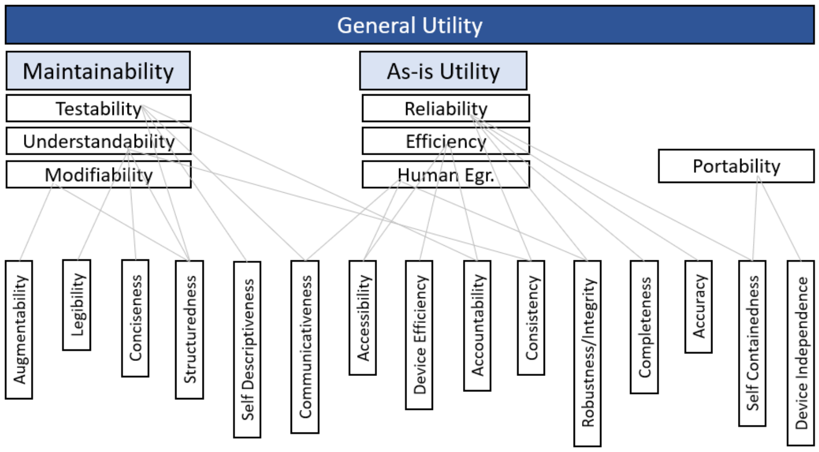


Figure 4
Software Quality Characteristics Tree from Boehm

Moving forward in time, the Rome Air Development Center (RADC) separated "quality attributes" into consumer-oriented vs. technically-oriented types [37], as seen in Figure 5. The former type is perceivable by the user and are thus aligned with non-functional user requirements. Quality aspects are categorized under performance, design, and adaptation.

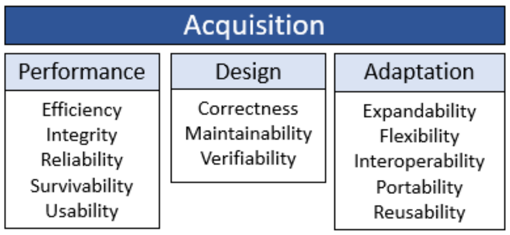


Figure 5
Customer-Oriented Quality Attributes from RADC

The system acceptability breakdown by Nielsen uncovers additional attributes related to user-perceived quality, with inclusions for utility and social acceptability [38]. See Figure 6 below.

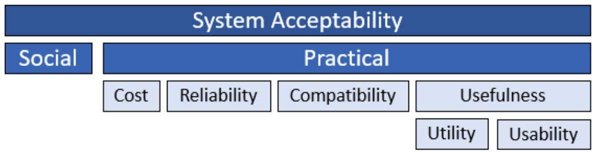


Figure 6
Taxonomy of System Acceptability from Nielsen

2.2.5.2 ISO 25010

The above historical quality models each provide a set of organized factors related to quality. Developed on the basis of these, the ISO 25010 model [39] (see Figure 7) has gained widespread use within the software community. Formally known as ISO 9126, it was brought into the ISO 25000-series as part of the SQuaRE (Software product Quality Requirements and Evaluation) family of standards. Related to HCD, it covers a wide array of user needs and provides a framework for product evaluation. Each first-level characteristic is decomposed into second-level attributes which are both measurable and verifiable. The specific quality metrics or other definable methods for evaluation are determined by the organization or project team.

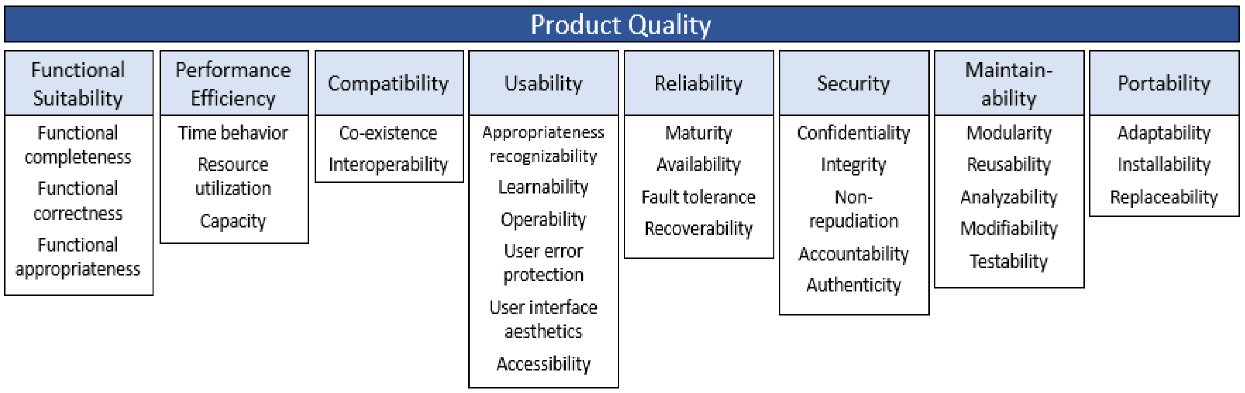


Figure 7
Quality Model from ISO 25010

These quality attributes cover the internal and external aspects of the system itself. Applying this model has also led to the complimentary definition of "quality in use" characteristics through adopting a systems view of quality [40], captured as: effectiveness, satisfaction, efficiency, safety, and context coverage. See Figure 8 below.

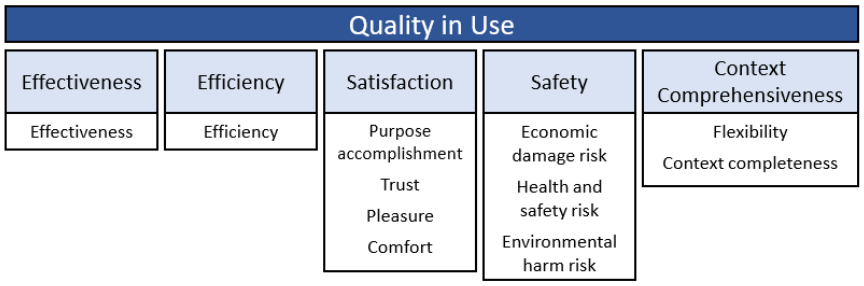


Figure 8
Quality in Use Aspects from ISO 25010

The above overview is intended to highlight the challenges in addressing the complex topic of user-related quality. It subsequently highlights the importance in leveraging a quality model in requirements development.

2.3 Engineering Processes

Going beyond a focus on requirements themselves, it is helpful to consider when and how they are incorporated into a project. Selections from the many processes that exist in research and practice are discussed below. Aspects which best correspond to URs are highlighted.

2.3.1 ISO 15288

The ISO 15288 standard on system lifecycle processes [32] covers several important aspects regarding the incorporation of user-related requirements. As previously discussed, URs can be classified as a subset of "stakeholder requirements". Figure 9 organizes the various technical processes [41] according to the systems development V-model [15].

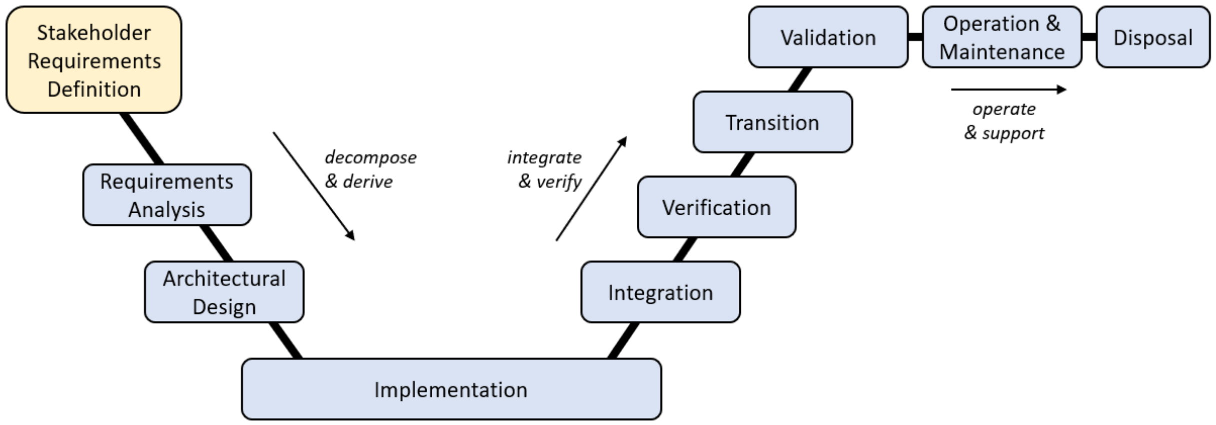


Figure 9
Technical Processes from ISO 15288

ISO 15288 describes stakeholder requirements as the clear and traceable representation of stakeholder needs and expectations. They form the basis for the development of system requirements, which are the technical solution to meet the user-oriented stakeholder requirements. They also are used directly as the basis for validation criteria, which determine that the "right product" is built from the stakeholder's viewpoint. These notions are adapted in Figure 10 below.

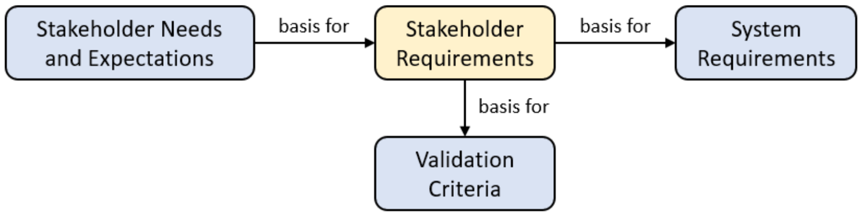


Figure 10
Stakeholder Requirements Relations from ISO 15288

Note, there exists a similar ISO standard specifically for software lifecycle processes [42], which also describes the same content for the purposes of this research.

2.3.2 ISO 29148

From a requirements-oriented viewpoint, the ISO 29148 standard on requirements engineering outlines an iterative process for requirements definition and their realization by a system design [33]. See Figure 11 below. It can be applied recursively at multiple layers of development. The recursive breakdown is also iterative throughout, such that no aspect is ever finalized during the overall effort.

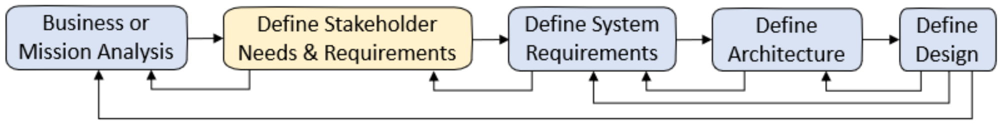

Business or Mission Analysis
Define Stakeholder Needs & Requirements
Define System Requirements
Define Architecture
Define Design

Figure 11
Design Process from ISO 29148

2.3.3 ISO 9241-210

Offering a slightly different perspective from the user's viewpoint, the standard ISO 9241 on human-centered design describes the key process steps when creating an interactive product [43]. This human-centered approach highlights the basic aspects while emphasizing the iterative nature of the process and the need for URs. See Figure 12 below.

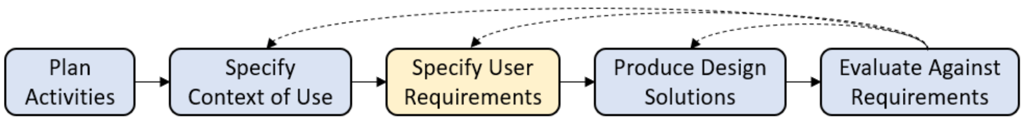


Figure 12
Human-Centered Design Process based on ISO 9241-210

2.3.4 SEBoK Process

SEBoK covers a process on defining stakeholder requirements from stakeholder needs [24], which adds detail to the stakeholder steps seen in ISO 15288 and ISO 29148. The list of process activities provides a useful context in understanding a viable approach to creating and managing the requirements. The process steps have been adapted into an illustration in Figure 13.

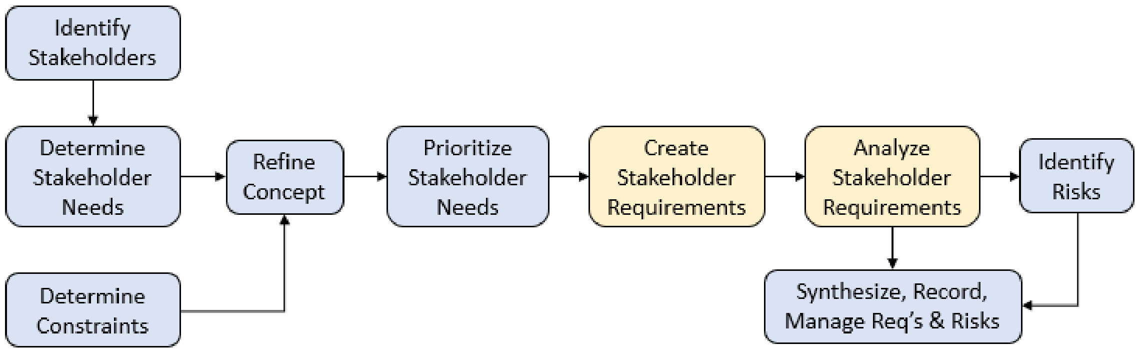

Identify Stakeholders
Determine Stakeholder Needs
Refine Concept
Prioritize Stakeholder Needs
Create Stakeholder Requirements
Analyze Stakeholder Requirements
Identify Risks
Determine Constraints
Synthesize, Record, Manage Req's & Risks

Figure 13
Process for Stakeholder Needs and Requirements from SEBoK

2.3.5 ANSI/EIA 632

The Electronic Industries Association 632 standard [44] describes overlapping engineering processes with some useful additions. First, user & customer requirements are decisively separated from other stakeholders, highlighting their importance. Second, the described process shows how the URs are included at the top layer of abstract development and influence the end-product through layered cascading and system decomposition. See Figure 14 below.

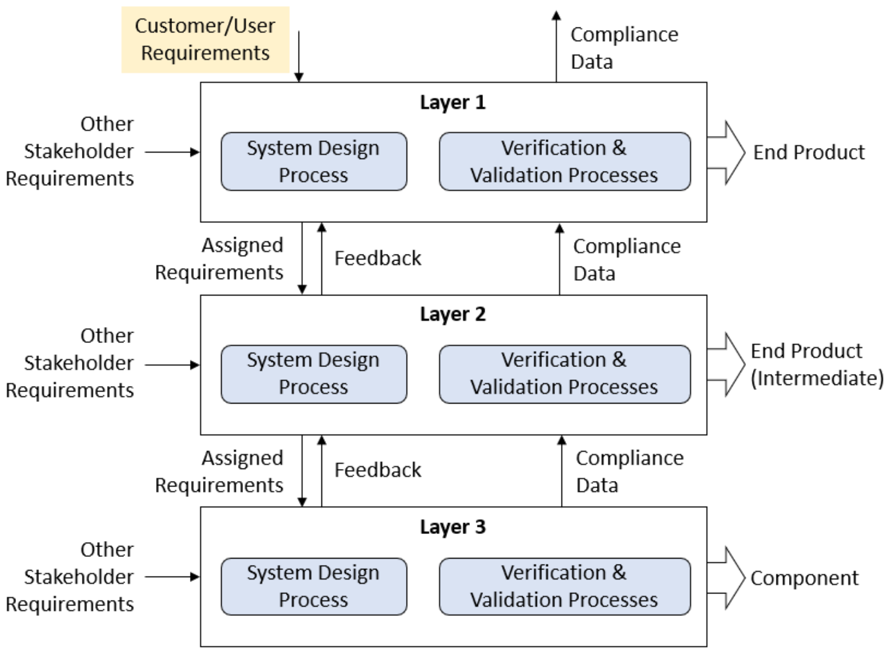

Customer/User Requirements
Compliance Data
Layer 1
Other Stakeholder Requirements
System Design Process
Verification & Validation Processes
End Product
Assigned Requirements
Feedback
Compliance Data
Layer 2
Other Stakeholder Requirements
System Design Process
Verification & Validation Processes
End Product (Intermediate)
Assigned Requirements
Feedback
Compliance Data
Layer 3
Other Stakeholder Requirements
System Design Process
Verification & Validation Processes
Component

Figure 14
Layered Engineering Process from ANSI/EIA 632

2.3.6 Agile Requirements

Finally, URs can certainly be applicable within Agile processes used in software development. As a brief overview, Agile is founded on a set of values and principles, described in the Manifesto for Agile Software Development [45]. These form the foundation upon which various approaches can be implemented. It advocates a strong focus on the end user and the product's user experience. These notions are well aligned to UR incorporation.

Various methodologies exist which support Agile requirements, such as Scrum [46], Kanban [47], and Extreme Programming [48]. Each method allows opportunities to incorporate a requirements framework throughout design and development [49]. Although one of the Agile values is to provide working software over comprehensive documentation, this does not exclude the need for high-level requirements at the stakeholder level.

2.4 Model-Based Systems Engineering

With growing system complexity, specifications and descriptions of systems become increasingly difficult to manage in documentation. Requirements management is no exception. As such, modeling is included here as a best practice in the field.

To briefly introduce the topic, MBSE leverages models to capture system information and can thus be used to replace the traditional document-based artifacts of systems engineering. It synthesizes several informational elements into a coherent model of the system, such as requirements, design, analysis, and testing [50]. Application of MBSE is commonly performed using the standardized Systems Modeling Language, or SysML [51]. SysML was created as an extension of the Unified Modeling Language from the object-oriented software engineering domain [52], in order to support general-purpose specifications of systems and systems-of-systems.

Using SysML, model data is stored in a model repository in the form of elements and relationships between elements. Along with data tables, the data can also be exposed graphically in various diagrams, as purpose-built views to illustrate the model data. Diagrams are used to show requirement trees, architectures, behavioral flow, and much more. Cross-cutting diagrams can be used to show the relationships between elements of different types (e.g., requirements satisfied by functions and verified by test activities).

2.4.1 Purpose and Benefits

To help understand the best practices in incorporating URs, it is important to discuss the value and benefits of MBSE [53], particularly for the field of RE:

- Improved traceability. Direct relationships are created between model elements, allowing for better data storage and retrieval. Understanding of data and collaboration are greatly supported. Forward-tracing (i.e. "how is this achieved?"), backward-tracing (i.e. "why is this here?"), and lateral-tracing (i.e. "what is this related to?") all carry substantial benefit.
- Improved quality. Through organization of data and information, design decisions are improved and more robust. Completeness of coverage is supported, along with stronger relevance of content.

- Improved reviewing. Different views of the same data are readily available, allowing various stakeholder viewpoints to be directly provided. Data organization provides easier retrieval during information discovery.
- Improved analysis efforts. By viewing requirements within a model, the context and related elements are readily understood and can thus be incorporated into the analysis.
- Improved maintainability and change management. Model elements are nested with direct relationships and are changed with direct access to nearest-neighbor elements (which may also be affected). The latest information is always available in the model and never out-of-date (as may occur with documents).
- Improved reuse. Data management supports shared usage across projects and systems.

2.4.2 Requirements Modeling

Requirements captured within a broader system model can directly trace to related elements. These relationships include the following 5 types [51], each with a corresponding SysML dependency stereotype.

- Derived relationships between two requirements (SysML's «deriveReqt»). Establishes a requirement hierarchy and provides forward and backward traceability across levels of abstraction.
- Verification relationship between requirements and their corresponding test activities (SysML's «verify»).
- Refinement relationship between a requirement and an element with less/more specified detail (SysML's «refine»).
- Satisfaction of requirements by activities or structural elements (SysML's «satisfy»). Exposes which behavior or structural element is used to meet a requirement.
- General traceability to other model elements which are deemed as related (SysML's «trace»).

Generic examples are shown in Figure 15 below to illustrate the relationships in a SysML model.

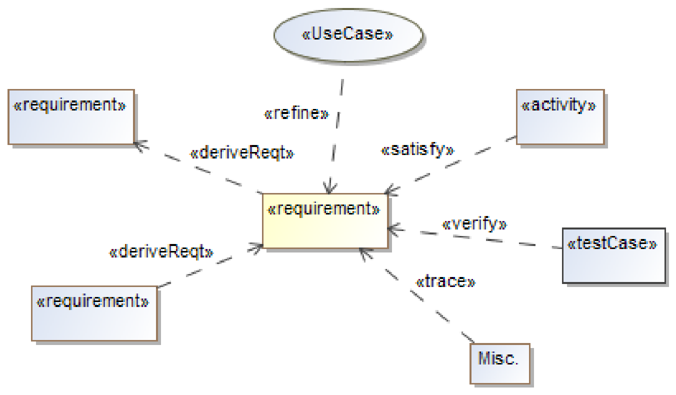


Figure 15
SysML Requirement Relationships

Modeling also enables the ability to capture properties of the requirements themselves in an organized fashion (beyond simply the "text" field of a requirement element). These properties are becoming more essential within the requirements engineering community in light of growing complexities and interdependencies of systems.

2.4.3 Goal Modeling

In comparison to textual requirements, human-centered content can also be incorporated into a system model through a variety of goal modeling techniques. Many of these methods may be considered as competitive approaches to the incorporation of URs, or as complimentary when used in conjunction. Many of the benefits offered by RE are arguably lost in these approaches, such as the ability to readily use URs as the basis for both the system design and validation testing, and to enable straightforward traceability. Therefore, it seems prudent to consider some form of URs in addition to goal modeling whenever it is applied.

Many methods exist to model goals as a part of an engineering effort, including Use Cases, NFR Framework, Tropos, i*, and GRN [54]. These techniques are worth exploring as the overlap and connection to URs is particularly important if URs are to be traced properly into a larger descriptive model. Several popular methods are described below.

2.4.3.1 Use Cases

Use cases have become a key method for modeling user needs and goals, facilitated by the introduction of UML software modeling [52]. Each use case represents an independent goal of the user, and can be decomposed into sub-goals via activities or other use cases, covering various levels of abstraction [55]. An example UML Use Case Diagram is shown in Figure 16 for an ATM system, with the primary users on the left side and their associated use cases.

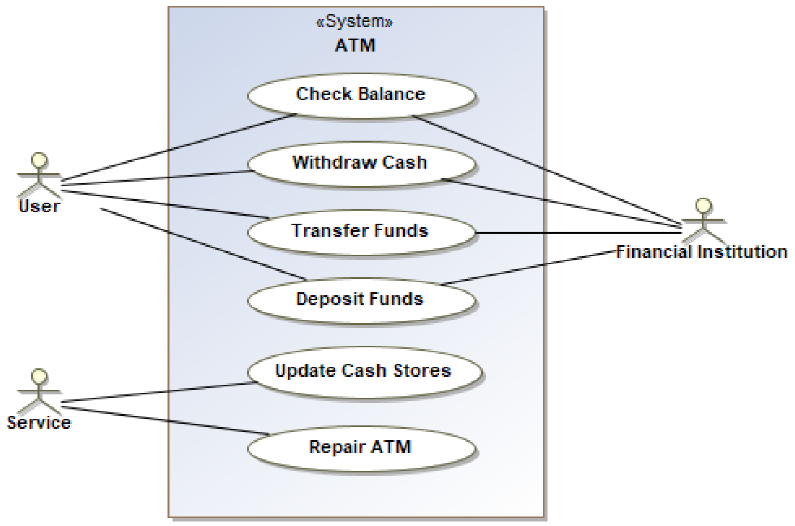


Figure 16
Example Use Cases for an ATM

A use case can also be associated with a descriptive use case specification, containing details such as the preconditions, steps involved, and post conditions. These details may readily tie to URs.

2.4.3.2 Goal-Oriented Requirement Language

The Goal-Oriented Requirement Language (GRL) is a common goal modeling notation in practice, which creates a taxonomy between hard goals (i.e. clearly verifiable goals), soft goals (i.e. not clearly verifiable), tasks (i.e. ways to perform or meet a goal), resources (i.e. a necessary physical or informational element), beliefs (i.e. rationale or assumptions regarding a goal), and key performance indicators (i.e. evaluation criteria). This modeling approach allows for human-centered content to be modeled in an organized fashion, giving the context for a goal and how it is met. However, it may limit integration to a comprehensive system model, and could easily exist as a standalone artifact without direct traceability to the system design. It is therefore recommended for integration with a more comprehensive modeling language such as SysML [56].

2.4.3.3 User Requirements Notations

The International Telecommunication Union introduced the User Requirements Notation (URN), which combines the GRL and use cases into a semi-formal approach to modeling both goals and scenarios in a single method [57]. In this way, it similarly supports the elicitation of URs. Similar to GRL, integration with SysML would allow for a stronger overall approach to modeling [58].

2.4.3.4 Scenario Modeling

Scenario modeling is a growing field, as scenarios carry an ease of understandability across stakeholder groups and can be readily leveraged for further downstream modeling efforts [10]. A well-defined scenario covers several important aspects in a single definition (e.g., subject actor, context entities, user actions, system actions, results), thus serving as a cross-cutting modeling technique which may elicit multiple URs. A recent example of such an approach is the PEGASUS method from the automotive industry [59].

2.5 Literature Review Summary

To conclude the literature review, several subject matters were discussed, each related to URs. These topics work together in a complimentary fashion to offer context and provide input to the UR proposal in the following chapter.

A variety of user-related information content was discussed. It was considered as input content towards the development of URs, as well as complimentary. Many approaches were briefly discussed, including user research methods, user stories, and HTA.

Perspectives on RE were given, including the separation of functional vs. quality requirement types. URs were noted as a form of stakeholder requirement, with guidance for URs described by ISO and NIST. Categories of quality aspects were also shown via quality models, as a reference to creating useful quality-URs. Requirements processes were also reviewed, with a focus on high-level URs.

Finally, a brief overview to MBSE was provided, enabled by the language SysML. Existing methods were described for requirements and goal modeling.

Upon reviewing the literature, the need for a UR framework and corresponding process is exposed. Each topic addresses a related aspect, but a synthesized form does not exist to allow for the incorporation of model-based URs into a product development effort. This thesis attempts to address this need, while leveraging the existing literature to the extent practicable.

Chapter 3. User Requirements Proposal

A framework for URs was developed along with a corresponding process for the incorporation of it into a product development lifecycle. The proposed work was developed based on MBSE, ISO standards, established practices within the Case Organization, feedback from subject-matter experts, and practical considerations (such as preventing over-complexity).

The sections in this chapter cover the following:

- The UR framework, capturing how URs relate to neighboring information
- Guidance on the syntax and usage for the URs
- A recommended process for early-phase incorporation of URs

3.1 UR Framework

As a method to incorporate URs into a project, a UR framework was developed to meet a set of goals, through leveraging ISO 15288 and SysML modeling. The details of its development into a viable framework are discussed in the following sections.

3.1.1 UR Framework Goals

The UR framework was strategically designed to meet the following goals:

- To provide direct traceability of URs to related artifacts, including:
    - Forward traceability of URs to system requirements and other design elements
    - Backward traceability of URs to use cases, validation attributes, and other high-level artifacts
    - Lateral traceability of URs to validation test plans, user-related analyses, and other relevant data
- To provide an understandable organization of URs within a system model, retaining sufficient simplicity to support ease of management and maintenance
- To include requirements attributes in order to enable the addition of URs into an existing engineering workflow

- To support the ease of incorporation of the framework into existing engineering practices at the Case Organization, as well as into other common practices in industry

To achieve the above, several sources worked together to serve as the basis for the framework development. SysML-enabled MBSE and ISO 15288 formed the foundation of the work, to support a formal organization of elements and to allow the framework to be incorporated into existing system models. The UR elements themselves were based on ISO 25065 on user requirements, which also motivated the inclusion of certain relationships to related informational entities. Finally, existing SysML elements and practices used at the Case Organization provided a degree of refinement to the overall framework. This brought it into a more precise form, to become something ready for incorporation into an existing workflow.

3.1.2 Incorporating ISO 15288

The ISO 15288 standard [32] relates that stakeholder needs and expectations serve as the basis for stakeholder requirements. Stakeholder requirements in turn serve as the basis for the system requirements and validation criteria. The standard also provides guidance that these relationships should maintain bi-directional traceability, which aligns well to a model-based approach.

Adapting this guidance to the user (instead of the more general “stakeholder”), the relationships are illustrated in Figure 17. This basic structure serves as the foundation for the proposed UR framework in the section to follow.

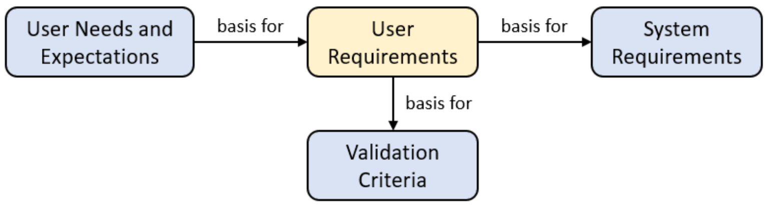


Figure 17
User Requirements Relationships from ISO 15288

3.1.3 Model-Based UR Framework

The complete UR framework is shown in Figure 18 below, illustrating the UR and its related elements within the context of a descriptive system model. Each relationship corresponds to a type of SysML dependency. SysML stereotypes are intentionally omitted for each entity in order to allow the framework to be adapted into custom SysML profiles. For example, a UR could be captured under the generic «Requirement» stereotype, or a custom «User Requirement» stereotype could be created.

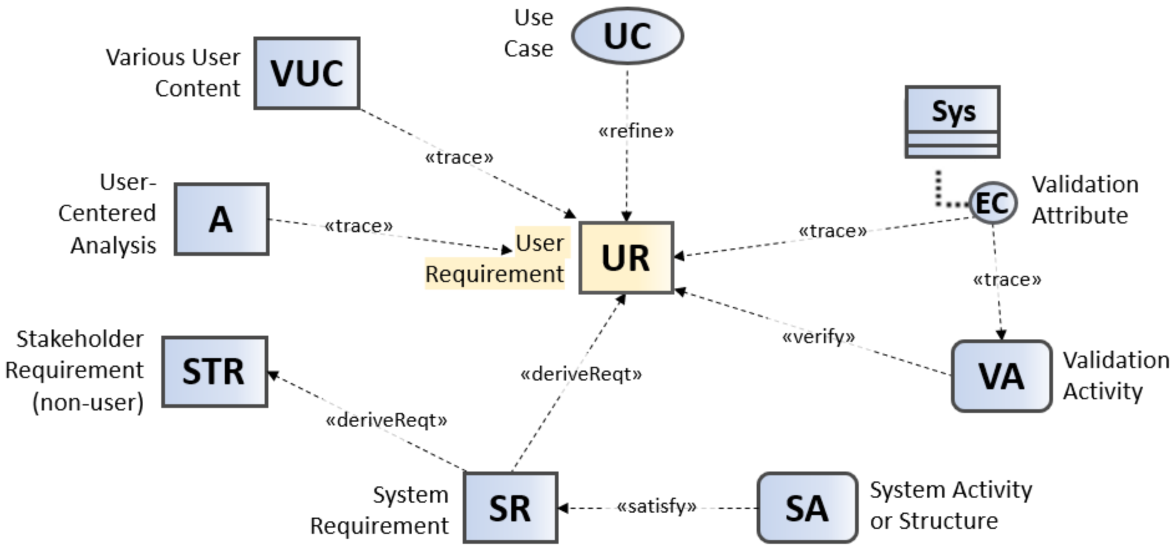


Figure 18
Model-Based UR Framework in SysML

There are certainly many other elements typically included in a system model which are not shown above in Figure 18. As such, this framework could be easily extended to include additional elements as deemed relevant by the modeler and project team. For instance, a business requirement may be captured as a parent element to a UR, a system function may be traced from a functional-UR, and so on.

The box for "Various User Content" is generic and serves as a catch-all for the many different forms of user-related information that may be a part of a system conceptualization process. Artifacts may exist in the form of user research findings, user stories, journey maps, etc. The particular information captured here is quite dependent on the type of system being built, the organization performing the work, and the process being employed.

The representation for "Validation Attributes" is depicted as a property of a class element (shown as the system itself). This approach corresponds to a suitable method for modeling this type of information in practice. Any class element (such as the system) is able to own properties related to its performance. These properties can be effectively used as the validation attributes, and often correspond to quality characteristics.

3.1.4 Requirement Properties

The URs are captured as SysML requirements elements, which carry certain properties to store the requirement details. The SysML specification [51] refers to three common properties of requirements elements: ID, Name, and Text. Additional relational properties expose the elements that have a direct relationship to the requirement: traced elements, parent requirements (i.e. derived from), derived requirements, satisfying elements, and verifying test cases.

Other properties for consideration include a prioritization or risk classifier, requirement type or category, workflow status, rationale text, and notes. Additional properties may also be needed for databasing and linkages to other tools (such as project management software). Ultimately, the list of properties depends on what is deemed useful by the organization for its requirements engineering process.

An example SysML profile for URs is shown in Figure 19. Certain properties are inherited from the more general «AbstractRequirement», and some properties are exposed as enumerated lists. This is meant to be an example of a viable modeling profile structure.

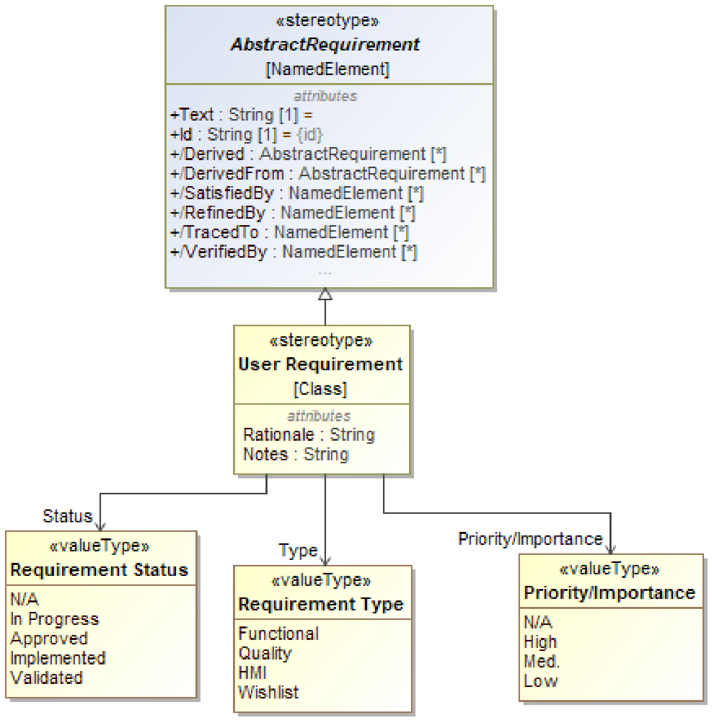


Figure 19
User Requirements SysML Profile

3.1.5 UR Framework Example

The following example in Figure 20 shows the traceability of a UR to its related elements using the previously described relationship types. A mix of standard and custom stereotypes is chosen for the elements. Without being constrained by the example shown, the framework could be applied for any type of UR, and at different levels of abstraction.

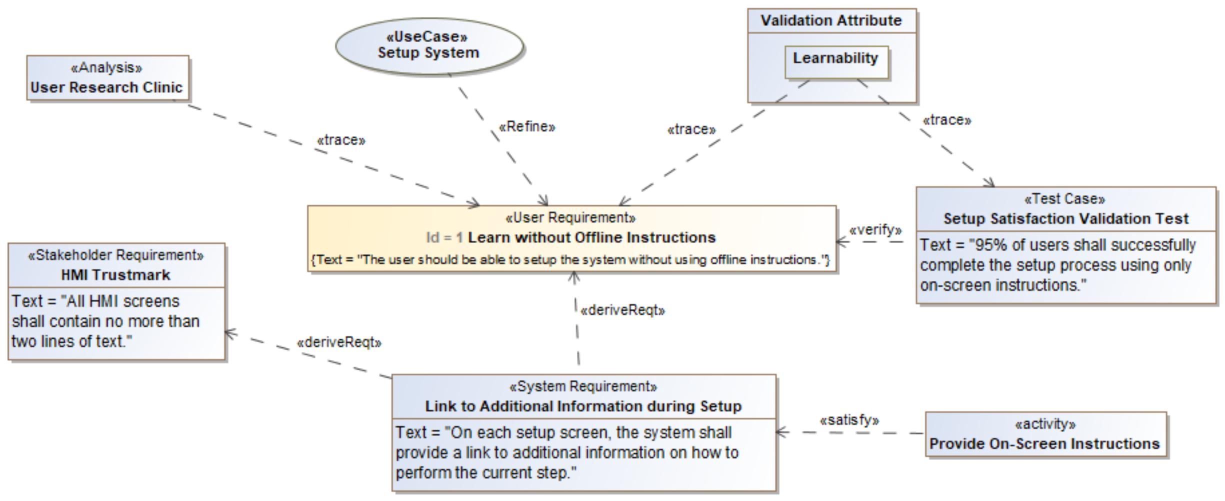

«Analysis»
User Research Clinic
«UseCase»
Setup System
Validation Attribute
Learnability
«trace»
«Refine»
«trace»
«trace»
«User Requirement»
Id = 1 Learn without Offline Instructions
{Text = "The user should be able to setup the system without using offline instructions."}
«verify»
«Test Case»
Setup Satisfaction Validation Test
Text = "95% of users shall successfully complete the setup process using only on-screen instructions."
«Stakeholder Requirement»
HMI Trustmark
Text = "All HMI screens shall contain no more than two lines of text."
«deriveReqt»
«deriveReqt»
«System Requirement»
Link to Additional Information during Setup
Text = "On each setup screen, the system shall provide a link to additional information on how to perform the current step."
«satisfy»
«activity»
Provide On-Screen Instructions

Figure 20
User Requirement Framework Example

3.2 Guidance for UR Syntax and Usage

Incorporating the UR framework is of limited value without guidance on how to create the URs that reside within it. Towards this goal, the textual writing of the requirements is addressed in this section, as well as how to elicit valuable URs for both functional and quality aspects of a system.

3.2.1 Incorporating ISO 25065

One of the foundational inspirations for this work arose from ISO 25065, which focuses on the creation of a user requirements specification [34]. This standard not only covers the importance and viability of URs, it highlights their relationship with the overall system development from a human-centered perspective.

Compared to the standard's guidance, several changes were applied to the syntax and usage of URs for this work. These modifications were made to account for model-based incorporation and to extend the usage of URs into a broader scope.

Firstly, regarding sentence structure, ISO 25065 begins UR sentences with the clause "With the system, …". This phrase was generally chosen to be omitted for the model-based URs. Using a model, the related entities of the requirement (such as the system itself) are captured via the relationships defined, and are therefore unnecessary within the requirement sentence itself. It would be otherwise redundant information (although not necessarily wrong to include if deemed valuable).

Secondly, this work replaces the verb "shall" with the word "should" in particular cases, such as for certain quality-URs without defined metrics, and for optionally-provided functional-URs. Doing so intentionally indicates a less clear-cut verification process, permitting an allowable disparity in the requirement's compliance by different users under various scenarios (whereas the word "shall" can infer that the requirement must be met in every case). It also reduces the liability for quality aspects that are met without defined metrics for their acceptance criteria. This approach also can serve as a mechanism to capture soft goals or optional functionalities that may or may not be met by a system depending on its development.

Finally, the ISO standard constricts the use of URs to only two categories of requirements (user-system interaction and use-related quality), excluding the following types:

- Requirements for non-user interacting humans (e.g., "the service technician shall be able to access the unit without removing other components")
- Requirements beyond direct interaction with the system (e.g., "the user shall be able to use the ticket for multiple fares")
- Requirements for functionalities (e.g., "the user shall be able to create customized buttons")
- Requirements on the content of use (e.g., "the user shall be able to operate the system at night")

The ISO group has reduced the scope of URs to meet their goals related to standards development and to manage its overlap with other standards [16]. For the needs of this particular work however, it seemed unnecessary to limit the usage of URs to necessarily exclude any of the above types. The spirit of the UR framework is to act as an aid or resource for the development of user-related content, and therefore does not provide an overly constricted boundary that must be adhered to.

3.2.2 Types of URs

For any complex system, a multitude of user-related needs, wants, and expectations exist. UR development deserves a degree of guidance in order provide adequate coverage of this space. Otherwise, ad hoc methods consume time and often miss important aspects.

As a way to guide their development, it is helpful to define the various categories for which URs could be written and allow requirement authors to consider each category accordingly. As discussed in Chapter 2, the types of requirements can first be separated into two categories: quality vs. functional. This breakdown can disaggregate further into additional layers of sub-types, described in the following section.

3.2.2.1 Quality-URs

Quality-URs are challenging to determine, as there are arguably an unlimited number of quality aspects that could be discovered and written as requirements. Perceived relevance becomes a key filter for their capture or omission. To support the discovery of relevant quality-URs, a quality model can be a useful resource to help explore commonly considered quality categories.

Given the widespread adoption of the quality model described in ISO 25010 [39], particularly in the software community, it is accordingly chosen to leverage here. However, some of the categories are not well-matched for URs but are rather more aptly suited for system requirements (e.g., the “maintainability” of a system). Also, many of the attributes are only relevant for certain systems and not for others. However, the quality model is meant be treated as a reference to help elicit URs, and any non-relevant categories can be ignored. Only URs deemed valuable should be captured for the particular system-of-interest.

It is recommended to carefully consider the quality categories individually for each use case or user task. Experience has shown it to be common to overlook one or more relevant aspects during the development of a complex system, and the quality model is designed to help mitigate this prospect. For example, consider a smartphone app which notifies a user whenever a motion sensor is activated near their house. This system includes potentially overlooked quality aspects such as “resource utilization” (does it significantly reduce the phone’s battery life?), “availability” (does it work at night when it is possibly most needed?), and many more.

The ISO 25010 quality model is shown in Figure 21, with separation between the characteristics for the product (including internal and external perspectives) and the quality-in-use for the product.

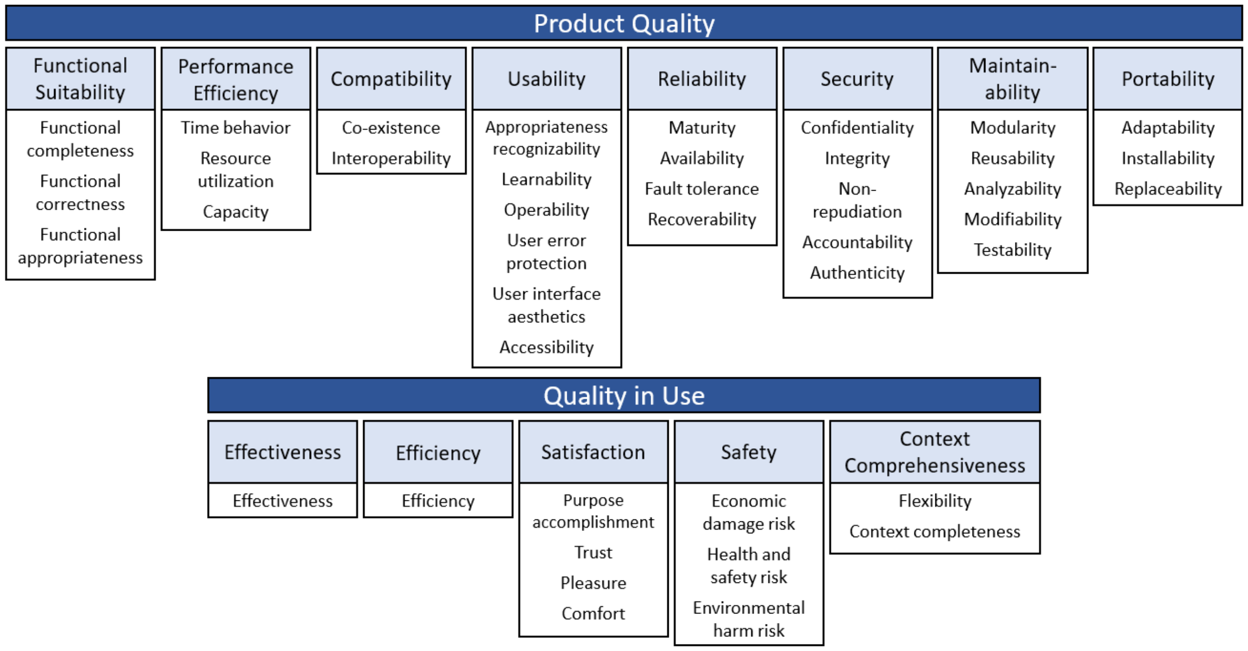


Figure 21
Quality Model from ISO 25010

3.2.2.2 Functional-URs

Functional-URs capture the functionalities (tasks, actions, or activities) desired by the user of a system. The functional-URs could tie to the different use cases of a system, or to the breakdown of a Hierarchical Task Analysis (ensuring at least one functional-UR is captured for each task). As such, it is recommended that a breakdown of functional sub-types be customized to a system’s specific behaviors, instead of attempting to create general-use categories (such as is done with a quality model).

One or more functional-URs could readily be written for each functionality identified. For example, if the user has an optional task to pause the system during use, a functional-UR could state, “During operation, the user shall be able to pause the system.” This is a purely functional requirement and does not cover quality aspects.

3.3 Recommended Process for UR Incorporation

The UR framework is better supported with an accompanying process for applying it on a project. As such, a viable workflow was established to suit the authoring and integration of URs into a development effort, leveraging ISO 15288, Agile principles, and practices at the Case Organization. The details of its development are discussed in the following sections.

3.3.1 UR Process Goals

While URs can be incorporated at any point during a project timeline (it can be said "better late than never"), there are endorsed methods regarding when and how to capture them for a more beneficial impact. Developing the process for incorporating URs included the following goals:

- To provide a placement of URs within the development timeline to directly influence the product design and development
- To ensure sufficient human-centered input content is available for creating useful URs
- To support compatibility with established processes, such as ISO 15288 and Agile development, and with those in practice at the Case Organization
- To allow flexibility for incorporation of URs into a variety of different engineering practices

A wide set of source material was used to develop general UR process guidelines to achieve the above goals. ISO 15288 process descriptions were used as the base material, followed by modifications and additions to customize around human-centered content. Agile principles were also included, along with current processes at the Case Organization. This led to a general UR process proposal suitable for use in a broad variety of applications.

3.3.2 Incorporating ISO 15288

The process for URs is aligned with the ISO 15288/12207 standard on system/software lifecycle processes, particularly the section on "stakeholder needs and requirements" [32, 42]. This section discusses the transformation of stakeholder needs into discrete text requirements. The user is considered as a particular type of stakeholder, and therefore UR development aligns well with this section in the standard.

The transformation activities are described below, paraphrased and adapted from the standard to capture the user as the stakeholder-of-interest. A degree of simplification is also enacted for succinctness.

1. Preparing for UR development
    a. Identify the relevant users of the system
    b. Identify a strategy for defining needs and handling tradeoffs
    c. Identify and obtain needed systems/services (e.g., requirements tools)
2. Capturing user needs
    a. Identify context of use
    b. Identify, filter, and prioritize user needs
    c. Capture details on user needs (including relationships)
3. Developing the concept
    a. Identify representative scenarios
    b. Identify user interactions
4. Developing user requirements
    a. Identify solution constraints
    b. Identify user requirements (according to concepts, scenarios, interactions, constraints, and quality aspects)
5. Analyzing user requirements
    a. Analyze user requirements (e.g., check for consistency, correctness, etc.)
    b. Identify critical performance measures
    c. Present requirements to users and resolve issues

The above process is further simplified and illustrated in Figure 22.

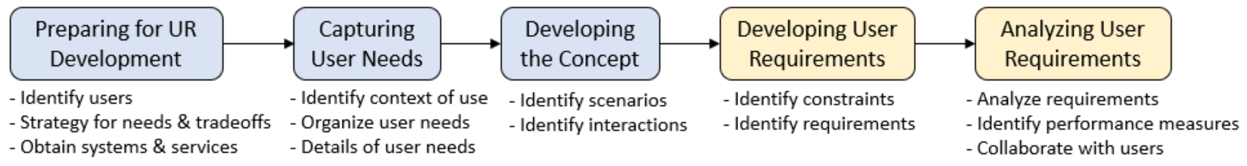


Figure 22
UR Process Adapted from ISO 15288

3.3.3 Incorporating Agile Development

Agile principles include the regularity of iterative change, a strong focus on the user, and direct collaboration with the user [60]. These notions themselves do not espouse a particular process, but rather provide a strong influence on the chosen or designed process to be employed.

The disposition of the UR proposal is aligned with the Agile principles. More specifically, incorporating URs into a product development process supports Agile in the following ways:

- URs become artifacts to directly capture user needs and expectations
- URs become a viable avenue to support customer collaboration and negotiation
- By including URs as a parent layer to system requirements, URs can represent as a more stable set of goals and criteria, allowing system requirements to be frequently iterated upon
- Early incorporation of URs supports a user-oriented system to be created and validated during initial development, reducing the likelihood for rework

3.3.4 Proposed UR Process

A general UR process was defined for the purposes of this study, (a) to act as a reference for UR guidance, and (b) to support the incorporation of URs at the Case Organization. It combines the concepts discussed above along with additional information on other aspects, such as user research, business needs, and project management content. For illustration purposes the process is shown as linear in Figure 23, however it is typically quite iterative in practice.

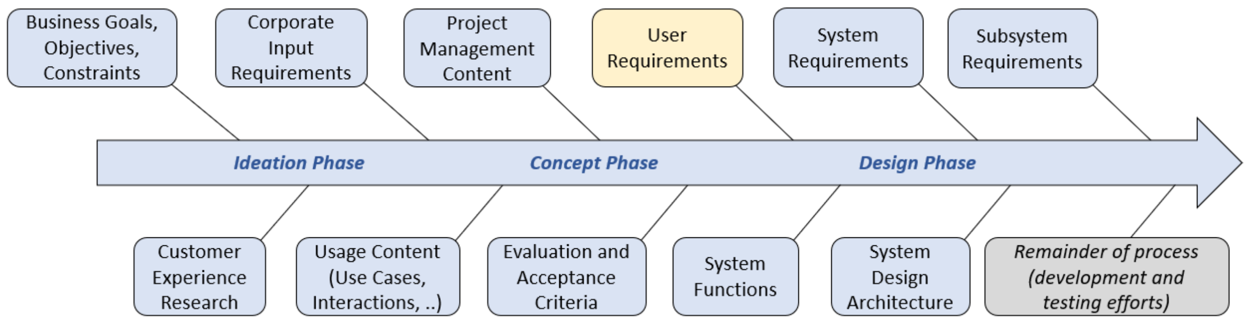

Business Goals, Objectives, Constraints
Corporate Input Requirements
Project Management Content
User Requirements
System Requirements
Subsystem Requirements
Ideation Phase
Concept Phase
Design Phase
Customer Experience Research
Usage Content (Use Cases, Interactions, ..)
Evaluation and Acceptance Criteria
System Functions
System Design Architecture
Remainder of process (development and testing efforts)

Figure 23
Proposed UR Incorporation Process

The remainder of the system development process following UR authoring is abridged, as the various methods for this phase are not important to define or distinguish for the purposes of this research.

3.3.5 Other Viable Processes

A single or strict process is certainly not necessary to incorporate the UR framework or portions of it. The above process depiction is intended to highlight a workable approach and is particularly suited for the Case Organization as a part of this research. It is worth noting that the UR framework fits into a larger process but does not drive the overall engineering effort. It should therefore be perceived and treated within this broader context.

Tailoring of the UR framework into the existing process of an organization is expected and encouraged. With or without URs, processes could be adapted using the following principles:

- Capturing clear and relevant human-centered aspects as discrete traceable objects
- Allowing user content to become the basis of the system design and the validation testing
- Managing functional vs. quality concerns in compliment of each other
- Leveraging the quality model to capture relevant quality aspects of a system

Chapter 4. Research Methodology

The proposed framework and process from the previous section was developed using the best available contemporary sources and practices within MBSE and product development. However, it remained unevaluated in practice at the time of conception, and there was no relevant literature found on the measured or expected impact of URs in industry. It was therefore deemed important to assess the proposal through multiple channels of available resources.

4.1 Research Context

All research was conducted at a large automotive company (the workplace of the investigator), referred to as the Case Organization throughout this document. MBSE was already widespread in its adoption at the Case Organization, along with Agile development practices and HCD methodologies. The setting was therefore deemed highly appropriate for a study on model-based URs.

This research study combined data from two primary sources at the Case Organization. The first workstream used a survey to gather impressions from various engineers on their perceptions of the UR proposal. The second effort involved an actual application of the UR framework into active projects as part of a design iteration, followed by a study on the consequent impact to the system design and testing plans.

For the survey, feedback was sought from engineers and analysts from the research, product development, and user experience departments. Most participants were actively involved in project work for new customer interactive vehicle features, typically involving vehicle automation. Participants were invited based on their involvement in user interactive projects.

For the project study, three technology projects were selected to incorporate the UR framework. These were chosen as projects which carried sufficient user interaction to deliver meaningful results and were at a suitable stage of development for UR application.

The Case Organization and all participants are kept anonymous to prevent any internal or external influence as a result of the published research, and to ensure separation from discussions made within this document. Also, the technology being worked on by the Case Organization is of a proprietary nature and is therefore not disclosed or included in examples.

4.2 Survey Study

This section describes the survey given to a set of engineers at the Case Organization. It represents the first half of the research effort (the second half being the project study described in a later section). The full survey can be found in the Appendix.

4.2.1 Overview

A self-directed survey was used to collect feedback on the proposed UR framework and process. Impressions were gathered as evidence on the expected impact for incorporating URs into a project. Open feedback was also received on areas of improvement and suggestions for additional work.

4.2.2 Research Questions

Several research questions were evaluated through the survey, organized below with their corresponding null hypotheses (H_0), which are numbered (i.e. $H_{0x)}$:

- Would URs benefit a system development effort?
    - H_{01}: URs would not provide benefit to system development.
    - H_{02}: URs provide equal benefit to different engineering efforts (concept, design, analysis, testing, implementation)
- Would URs benefit particular aspects of a system?
    - H_{03}: URs provide equal benefit to different system aspects (functionalities, quality, HMI)
- Do URs only benefit complex interactive systems?
    - H_{04}: URs provide equal benefit to systems with differing levels of complexity.
    - H_{05}: URs provide equal benefit to systems with differing levels of user interaction.
- Are certain types of URs more valuable than others?
    - H_{06}: All types of URs are equally important (functional, effectiveness, efficiency, satisfaction, wishlist)

- Is the UR framework easy to incorporate into a workflow?
    - *H_{07}*: The UR framework is challenging to incorporate.
- Are UR traceability relationships important?
    - *H_{08}*: It is not important for URs to have traceability to related elements.
- Which UR traceability relationships are important?
    - *H_{09}*: UR trace relationships to various elements are equally important (high-level content, use cases, system requirements, validation tests).

4.2.3 Survey Questions

A total of 34 survey questions were created against the above hypotheses, organized under 10 master questions (see Appendix). Microsoft Forms provided the interface through a standard web browser, secured by the Case Organization. This method achieved an overall readability and an understandability without explicit instructions on how to complete the survey, allowing it to become self-directed.

The majority of the survey items (28 of 34) were presented as fragmented statements with a corresponding selectable Likert item. These items measure the level of agreement or disagreement to the target statement on a psychometric scale [61] with successive verbal labels between opposite end points.

The following 5-point scales were used:

- Agreeability: [Strongly disagree, Disagree, Neutral, Agree, Strongly agree]
- Importance: [Not at all important, Slightly unimportant, Moderately important, Very important, Extremely important]

In addition to the Likert items, the survey began with a selection on the years of experience in various areas of engineering (overall, requirements, MBSE, product design, HCD, verification/validation testing), and concluded with open-ended questions to gather general feedback on the proposed framework and process.

4.2.4 Pilot Testing

Before distribution, the survey questions and format were evaluated by two university advisors and three subject-matter experts at the Case Organization. Representation was sought from systems engineering, requirements engineering, and HCD. The following review feedback was subsequently implemented:

- Established a stronger connection between the survey statements and the research questions
- Clarified syntax of the survey statements
- Re-ordered questions to improve flow
- Removed or reworded ambiguous content

The pilot testing also served to estimate the time to complete the survey and to gather general impressions. The comments from the reviewers affirmed the survey's quality and suitability.

4.2.5 Survey Distribution

The survey participants resided in the user experience, research, and product development departments at the Case Organization. As a precondition to participation, invitees needed a familiarity with requirements, HCD, and model-based methods. Given these constraints, efforts were made to gather representation across several related work areas within engineering, from the following groups:

- Requirements engineering
- Customer experience
- Verification & validation testing
- Human factors
- MBSE
- Interactive software development
- Product design

Possible candidates were identified based on their active involvement in one or more of the above areas within an ongoing technology project using RE and MBSE. Through project rosters and networking, a list of 81 qualified individuals was compiled for an invitation to participate.

The survey was distributed to the identified candidates through an invitation email, along with the pre-survey materials and an informational consent document describing the details of involvement. Survey participation was voluntary and self-selected through the survey link within the email, leading to 59 responses.

4.2.6 Pre-Survey Materials

A presentation of slides was reviewed by each participant before taking the survey, which provided the necessary overview on the UR content. The material covered the critical aspects of the model-based UR framework and process, including:

- An overview of URs
- Rational for incorporating URs
- UR syntax patterns
- UR traceability framework
- Types of URs (including quality model)
- UR process proposal
- Example URs

Specific details were included in the slides to properly contextualize the UR proposal specifically within the Case Organization. This included the usage of specific and custom SysML stereotypes used at the company, popular terms with a company-wide vernacular meaning, and several UR examples from a recently completed project familiar to the participants.

4.2.7 Survey Data Collection

Responses were automatically collected in real-time via the Microsoft Forms hosting tool as each survey was completed. 14 days after invitation, the survey was closed and considered complete. All data was compiled into an anonymous table to protect participant identification information.

4.2.8 Survey Data Analysis Method

Descriptive and statistical analysis was performed on the survey data to evaluate the significance of the findings and the corresponding probability of rejecting the null hypotheses. Non-parametric tests were selected to analyze the categorical data, using χ^2 and Kruskal-Wallis methods. These tests were chosen due to the ordinal data types and lack of expected normalcy [62]. The approach is discussed in more detail in Chapter 5.

4.2.9 Protection of Participants

All participant data was kept confidential throughout the study. Participant identifiers were deleted within 1 week of analysis, and the data was not shared. All data was maintained on a secure server at the Case Organization with password protection. Exempt approval from the IRB was received for the survey and its distribution, along with approval and cooperation by the Case Organization.

4.3 Project Study

This section describes the project study in which the UR framework was incorporated into several product development projects at the Case Organization. It represents the second half of the research effort, complimented by the survey study discussed in the previous section.

4.3.1 Overview

The project research involved implementing the UR framework into active projects which were not already using URs as a part of their development effort. Each project underwent a mid-cycle design iteration to introduce URs into their system models according to the UR framework, adding a new layer of requirements to the exiting artifacts. The system requirements and validation test plans were subsequently modified to accommodate the new UR content and achieve traceability. The changes caused by the UR inclusion were identified using a SysML model comparison, and each change was categorized to help understand its practical impact.

4.3.2 Research Questions

The purpose of the case studies was to collect evidence towards the following research questions:

- Does the incorporation of URs lead to changes in the system design?
- Does the incorporation of URs lead to changes in validation testing?
- Does the incorporation of URs lead to unmanageability of requirements data in a project?

These questions were evaluated by observing changes in system requirements and validation plans for each of the case-studied projects, after the URs were incorporated. Dependent variables were determined as (1) changes to the system design (functional vs. quality), (2) changes to validation plans, and (3) the complexity of the incorporated UR framework. Analysis also included consideration for the type of change observed, knowledge re-structuring, and the addition vs. alteration vs. removal of content.

4.3.3 Selection of Projects

Three projects were chosen for the study, each involving the advanced engineering of semi-automated driving systems. These were selected from the available projects at the Case Organization which met the following criteria:

- The system was user-interactive in nature (more than 3 user inputs)
- The system had sufficient design complexity (more than 5 functions)
- The workflow was performing iterative design cycles via an Agile approach
- MBSE was being used to create and maintain a descriptive system model
- No URs were written or planned for in the system model
- Validation test planning was included in the scope of the project
- The project team was willing to include the UR framework into their model

4.3.4 Incorporation of UR Framework

A set of URs was written for each project during a dedicated effort spanning approximately 2 weeks. See Figure 24 below. UR authoring was performed by a group comprised of members from systems engineering, human factors, and product design.

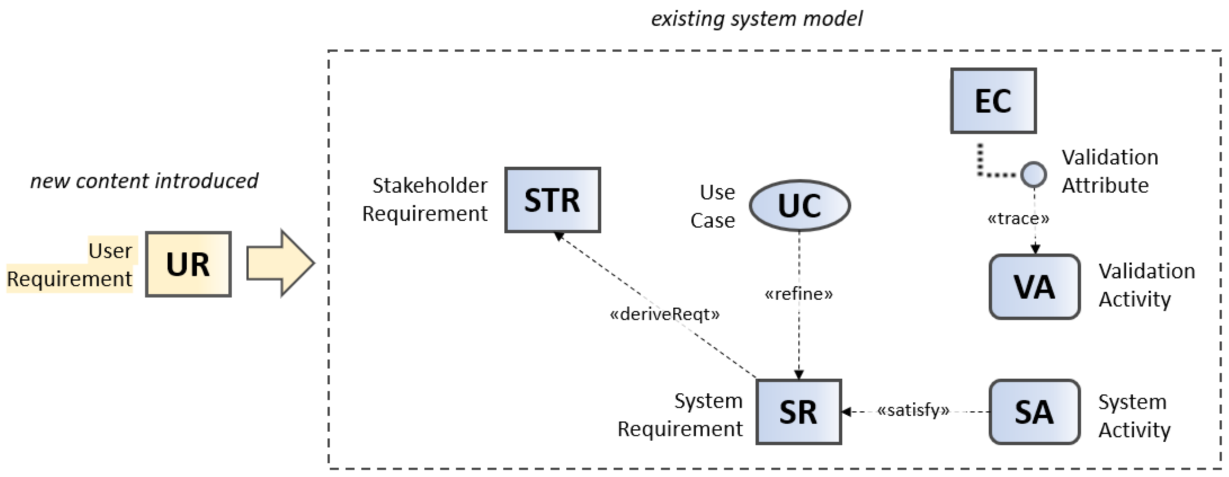


Figure 24
URs Introduced into the Existing System Model

URs were based on the existing use cases and latest concept specification for the system, which included a task analysis, stakeholder requirements, system activities, and other various user-related content. At least one functional-UR was written for each use case and/or user task.

Quality-URs were captured using the quality model as a reference guide, and relevant requirements were identified based on their perceived importance to the system operation.

Careful effort was made to not refactor the existing system requirements into URs, which would constitute a bottom-up effort and simply re-describe the engineering decisions in the form of URs. The UR framework is intended as a top-down human-centered approach, as opposed to describing system details in UR format (since these details may or may not serve the user, especially if written without human-centered methods).

4.3.5 Project Iteration

The set of URs was delivered to each project for integration into the system model. Instructions were given to the project team (see Figure 25) to trace the URs to use cases, to shift the system requirements as derived elements from the URs, and to update the validation testing plans to ensure coverage of all URs. Project teams were given the option to update the URs themselves if desired, to support seamless integration into their existing models.

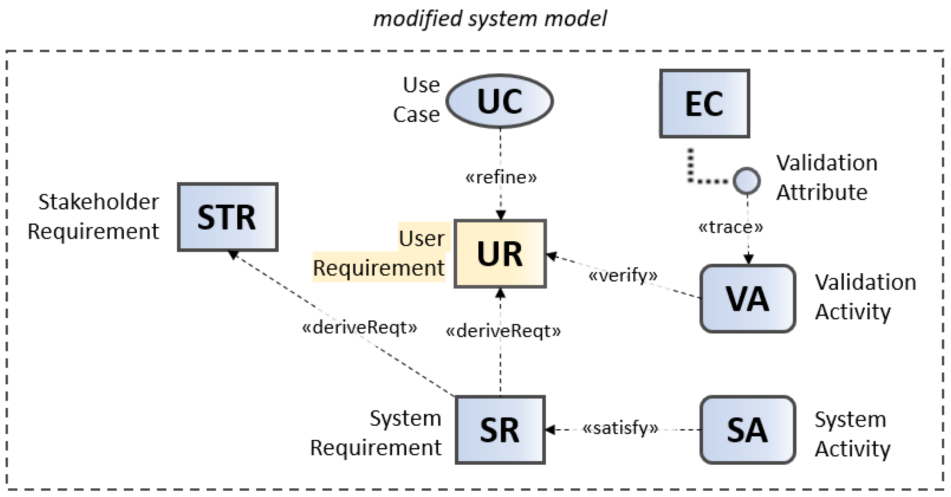


Figure 25
System Model after UR Incorporation

4.3.6 Project Data Collection

The design iteration was considered complete once the URs achieved traceability to the system requirements and validation plans. The data was subsequently harvested as a snapshot of each project's SysML model.

Each change to the system design and validation was classified using the following categories:

- Product change vs. documentation-only change (i.e. would the change affect the product itself, or just the documentation)
- Functional vs. quality change (i.e. new/modified/removed functionalities vs. changes to a testable quality aspect)

Additional data was collected on the total number of affected elements and relationships, in order to measure the informational load and model complexity involved in the UR framework inclusion.

4.3.7 Project Data Analysis Method

Given the limited sample size of only three projects, all data was descriptively analyzed. Changes were evaluated based on their influence on the system design to determine insights regarding the practical impact of URs. Details are covered in Chapter 5.

4.3.8 Protection of Participants

All participation in the project study was considered part of the standard work activities within the Case Organization. The application of URs was deemed useful to each project and the case studies were able to occur as part of the value-driven development effort.

4.4 Research Method Summary

In summary, the research study involved two distinct channels: (1) a survey to collect impressions from qualified individuals on the expected impact of URs, and (2) a study of UR impact on three actual projects as part of a mid-cycle design iteration. Available resources were leveraged at the Case Organization to conduct the studies, in the context of advanced automotive technologies.

The scope is limited to a single company yet includes a breadth of exposure to several departments, engineering disciplines, and projects. It captures an appraisal for the expected impact alongside an evaluation of the actual impact of URs, harnessing insights from each to better understand how URs affect product development.

Chapter 5. Research Results

This chapter provides the results from the survey and the project case studies. Descriptive and statistical analysis is also performed to support interpretation and insights.

5.1 Survey Results

The survey is broken down into three components, covered separately in the sections to follow:

1. Participant characteristics: an assessment of the participant's work area and years of experience
2. Likert-type questions: investigating the participant's perceptions related to URs
3. Open questions: free-form feedback on the UR proposal

5.1.1 Participant Characteristics

A total of 59 participants responded to the survey (voluntarily self-selected from the 81 participants invited via email), with an average of 58 minutes taken per person to complete. The population included engineers, designers, and developers at the Case Organization. Each participant was asked to indicate their years of engineering experience in various work areas. Work experience in each category was compiled into Figure 26 and Table 2 below, illustrating the distribution across Requirements Engineering, Model-Based Systems Engineering, and Verification & Validation involvement.

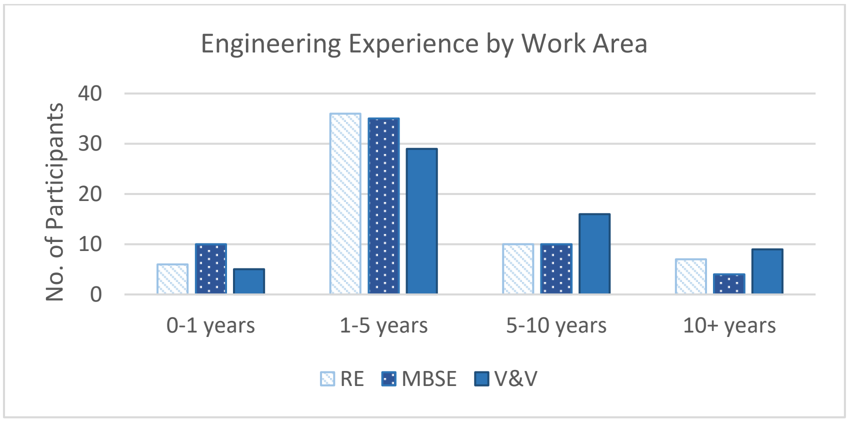


Figure 26
Engineering Experience Breakdown of Survey Participants

Experience Area	Average Years of Experience
Overall Engineering	10.3
RE	5.0
MBSE	4.1
V&V	6.1

Table 2
Mean Years of Experience of Survey Participants

5.1.2 Likert-Type Questions

The results from each of the Likert-type questions are shown in Figures 27-33. Actual responses are portrayed in charts, followed by statistical analysis results.

5.1.2.1 Data Visualization

The results from each Likert-type question are visualized using the Diverging Stacked Bar Chart [63]. As a modification from the 100% stacked bar chart, each bar is offset according to the centerline of neutral agreement. Bar length corresponds the response frequency.

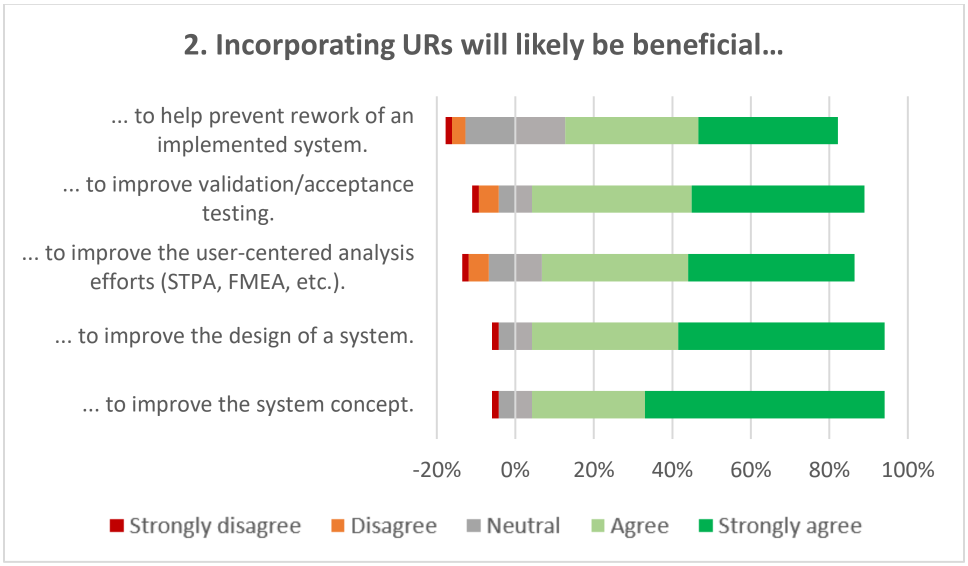


Figure 27
Survey Results for Question 2

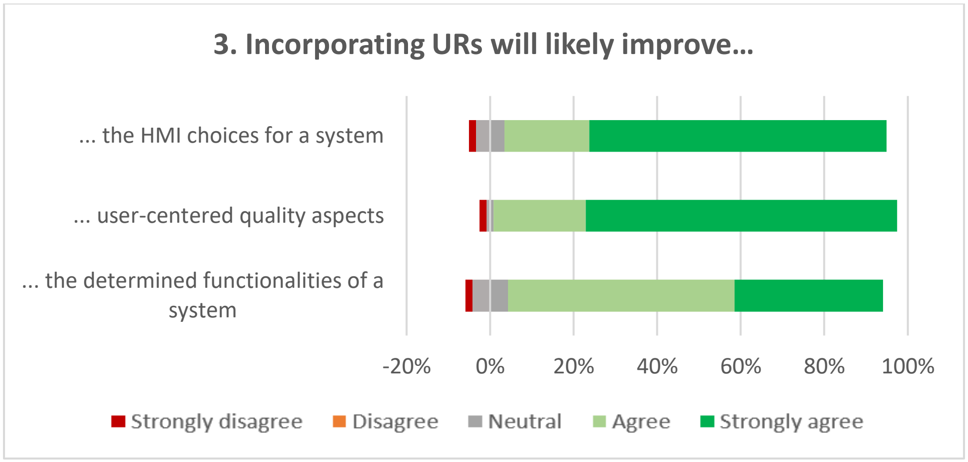


Figure 28
Survey Results for Question 3

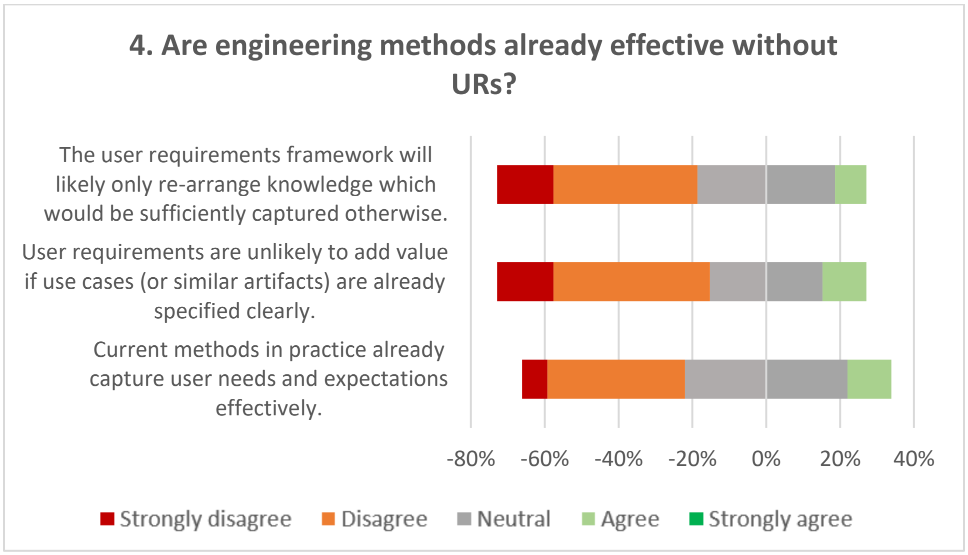


Figure 29
Survey Results for Question 4

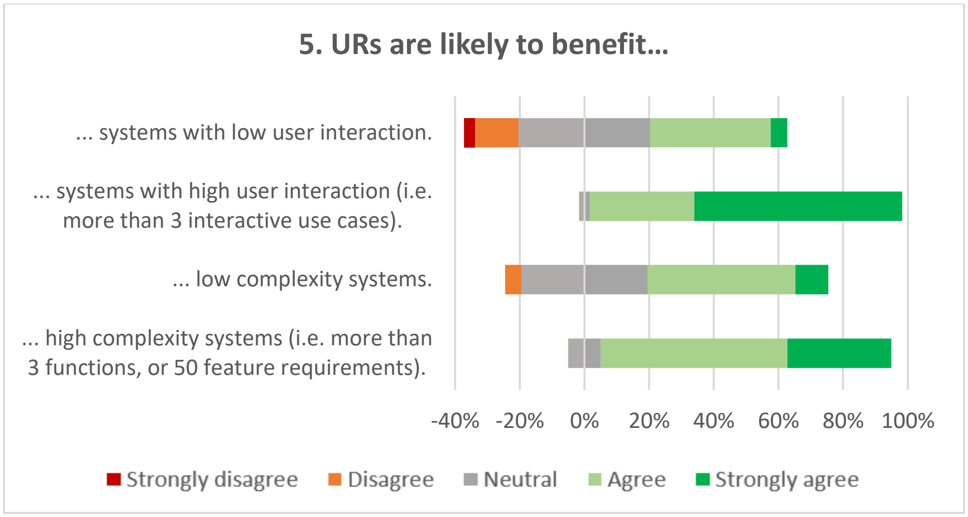


Figure 30
Survey Results for Question 5

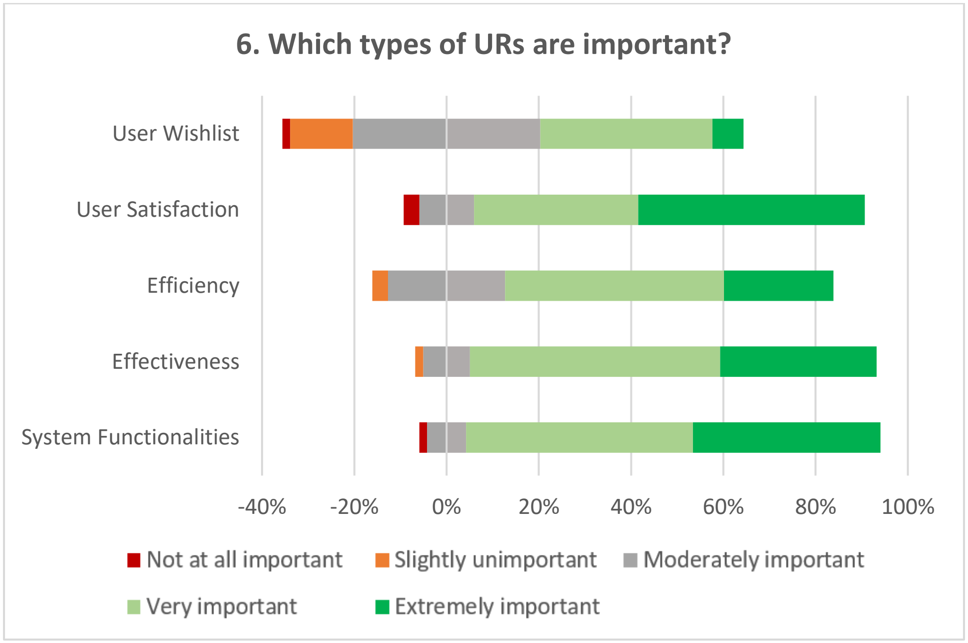


Figure 31
Survey Results for Question 6

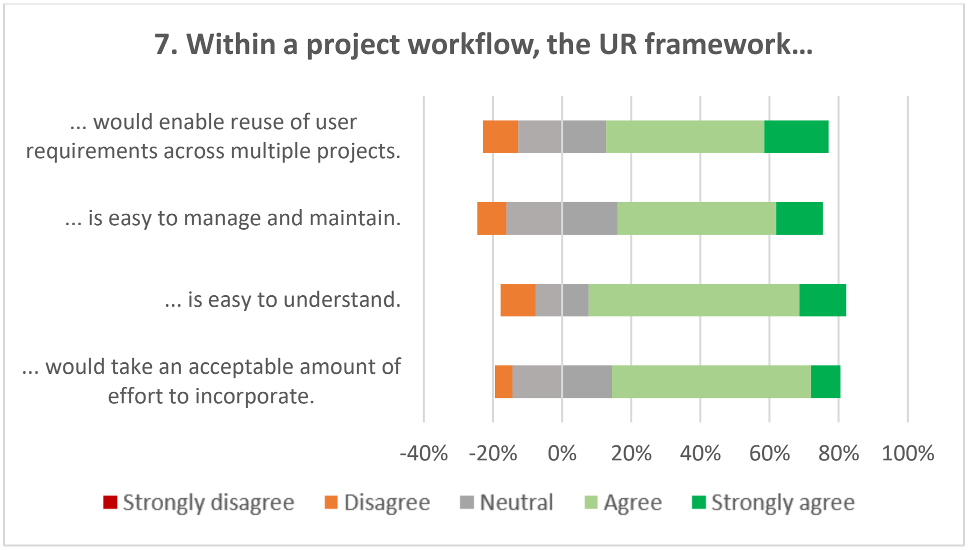


Figure 32
Survey Results for Question 7

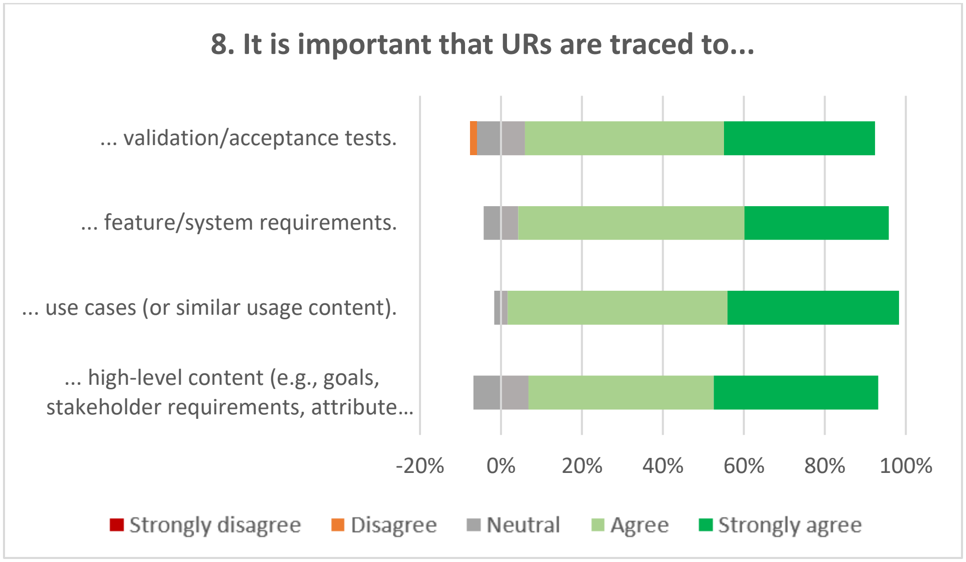


Figure 33
Survey Results for Question 8

5.1.2.2 Statistical Methods

To analyze the survey data, median scores and interquartile ranges (IQR) are used to support the ordinal categorical data types (in contrast to mean and standard deviation for numerical data). Statistical testing was performed using Chi-Squared (χ^2) and Kruskal-Wallis (K-W) methods, which were chosen to accommodate the categorical data and the lack of expected normalcy [62]. Several responses related to the research question were grouped together and aggregated into a single ordinal rating as evidence against the null hypothesis. Although the grouped data could have created a composite score for continuous variable analysis [64], it was deemed unsuitable to move away from ordinal data given the survey topics.

Each research question was analyzed through one these methods, using a combination of multiple related survey items. The method selection was made according to the following rationale:

- Median and IQR: To assess the overall perceptions of participants.
- χ^2 testing: To statistically evaluate the perceptions of participants against an expectation of neutral responses.
- K-W testing: To statistically compare factors within the same question against each other.

The detailed approach and results from each method are shown in the sections to follow.

5.1.2.3 Survey Data Organization

The survey questions, and their included Likert items, were written to address the research questions and their associated null hypotheses (see section 4.3.2). Each survey question served as evidence toward one or more of the null hypotheses and assigned based on their corresponding suitability. The type of statistical test for each hypothesis was chosen depending on whether the evaluation was designed for overall feedback vs. comparison of factors (χ^2 vs. K-W). The number of Likert items for the assigned survey question(s) were totaled as variable k to indicate the number of unique responses received. A sample size $n_T = 59$ was used for all questions (i.e. all questions were answered by all 59 participants). The assigned statistical test, applicable survey questions, and total Likert items are tabulated in Table 3 below.

H_0 No.	H_0	Statistical Test	Applicable Survey Questions	Total Likert Items (*k*)
H_{01}	URs would not provide benefit to system development.	Median & IQR, χ^2	Q2, Q3, Q5, Q6	17
H_{02}	URs provide equal benefit to different engineering efforts.	K-W	Q2	5
H_{03}	URs provide equal benefit to different system aspects.	K-W	Q3	3
H_{04}	URs provide equal benefit to systems with differing levels of complexity.	K-W	Q5	2
H_{05}	URs provide equal benefit to systems with differing levels of user interaction.	K-W	Q5	2
H_{06}	All types of URs are equally important.	K-W	Q6	5
H_{07}	The UR framework is challenging to incorporate.	Median & IQR, χ^2	Q7	4
H_{08}	It is not important for URs to have traceability to related elements.	Median & IQR, χ^2	Q8	4
H_{09}	UR relationships to various elements are equally important.	K-W	Q8	4

Table 3
H_0 Corresponding Survey Questions

5.1.2.4 Median Score and IQR Results

Descriptive statistics were used for analysis of questions on the overall impressions of participants related to the research question. Multiple Likert items were combined into a single grouping for each question, as shown in the previous section. Medians were calculated by finding the true median for each item (an integer between 1 and 5, corresponding to the five Likert-type options), and averaging these values to generate a composite median score for the research question. The interquartile range (the difference between quartile 3 and quartile 1) was similarly calculated to indicate the variation spread in the data.

Table 4 below shows the calculated medians and interquartile ranges. To help further understand the data, a summation of positive (agree + strongly agree) and negative (disagree + strongly disagree) responses is given, with neutral selections omitted.

H_0 No.	Research Question	"Agree / Strongly Agree" Count	"Disagree / Strongly Disagree" Count	Median Likert Score	Interquartile Range (Q3-Q1)
H_{01}	Would URs benefit a system development effort?	799	44	4.2	1.0
H_{07}	Is the UR framework easy to incorporate into a workflow?	156	20	4.0	0.9
H_{08}	Are UR traceability relationships important?	213	1	4.0	1.0

Table 4
Survey Results for Median and IQR

Median values averaged to a value of 4, corresponding to the "Agree" selection to the survey statements. Responses for "Strongly agree" were also the most frequent, occurring in 40% of the items. The second most common response was "Strongly agree", at a frequency of 39%.

The interquartile ranges are substantially low, indicating the participants' opinions are generally aligned (i.e. non-polarized) across the survey. It can be observed that the participants were resoundingly agreeable towards the research questions. The total percentage of "Agree / Strongly Agree" responses was 80%, compared to a 4% occurrence of disagreement and a 16% occurrence of neutral.

5.1.2.5 Chi-Squared Test Results

To draw statistical conclusions for certain research questions, the Pearson χ^2 Test for Homogeneity was used to compare the proportions from the survey. The results were reduced into total counts above and below the neutral Likert option, which was compared against expected values of equal distribution (half of the results above neutral, and half below neutral). If the calculated test statistic χ^2 was less than the table value of the χ^2 distribution (using p=0.05 and degrees of freedom equal to one less than the total Likert items), the null hypothesis was rejected, indicating the data set showed a significant deviation from a neutral average response. The test statistic was calculated using the formula:

$$\chi^2 = \sum_{i=1}^{n_T} \frac{(O_i - E_i)^2}{E_i}$$

where n_T is the number of samples (n_T=59 participants), O is the observation count of i, and E is the expected count of i per the null hypothesis.

The results for each null hypothesis are shown in Table 5 below.

H_0 No.	H_0	Deg. of Freedom (k-1)	χ^2-crit.	χ^2	p-value	Reject H_0?
H_{01}	URs would not provide benefit to system development.	16	26.3	693.1	<0.01	Yes
H_{07}	The UR framework is challenging to incorporate.	3	7.81	105.5	<0.01	Yes
H_{08}	It is not important for URs to have traceability to related elements.	3	7.81	210.1	<0.01	Yes

Table 5
Survey Results using χ^2 Test

It can be observed that the null hypotheses are significantly rejected in favor of expectations for UR benefit, ease of incorporation, and importance of traceability. These results emphasize the positive impressions on UR incorporation, as reported by the majority of survey participants.

5.1.2.6 Kruskal-Wallis Test Results

Questions with comparative categories were analyzed to determine if differences across the test factors were present, using a K-W test to compare the sets. This type of test is the non-parametric test corresponding to a one-way ANOVA. Using the K-W test, the rank score for each factor was used to calculate the test statistic H, which was compared against the χ^2 critical value (using p=0.05 and the corresponding degrees of freedom). If $H > \chi^2$, the null hypothesis was rejected, indicating the data sets were not equivalent. The H statistic was calculated using the formula:

$$H = \frac{12}{N(N+1)} \sum_{i=1}^{k} \frac{R_i^2}{n_i} - 3(N+1)$$

where k is the number of categories, n is the samples per category, N is the total observation count across all categories, and R is the rank score for each category.

The results are tabulated in Table 6 below, organized by each null hypothesis.

***H_0* No.**	***H_0***	***N***	**Deg. of Freedom (*k*-1)**	***H***	**χ^2**	***p*-value**	**Reject *H_0*?**
H_{02}	URs provide equal benefit to different engineering efforts (concept, design, analysis, testing, implementation).	295	4	10.7	9.5	0.03	Yes
H_{03}	URs provide equal benefit to different system aspects (functionalities, quality, HMI).	177	3	15.7	6.0	<0.01	Yes
H_{04}	URs provide equal benefit to systems with differing levels of complexity (high complexity, low complexity).	118	2	16.4	3.8	<0.01	Yes
H_{05}	URs provide equal benefit to systems with differing levels of user interaction (high interaction, low interaction).	118	2	52.3	3.8	<0.01	Yes
H_{06}	All types of URs are equally important (functional, effectiveness, efficiency, satisfaction, user wishlist).	295	4	42.6	9.5	<0.01	Yes
H_{09}	UR relationships to various elements are equally important (high-level content, use cases, system req.'s, validation tests).	236	3	1.2	7.8	0.75	No

Table 6
Survey Results using Kruskal-Wallis Test

The above results show a rejection of the null hypothesis for *H_{02}*, *H_{03}*, *H_{04}*, *H_{05}*, and *H_{06}*, concluding that the factors contain significant differentiation from each other. *H_{09}* received results in support of the hypothesis, indicating factor alignment.

For the rejected hypothesis, the comparison is visualized in the following charts (Figures 34-37), using the calculated rank score *R* for each factor.

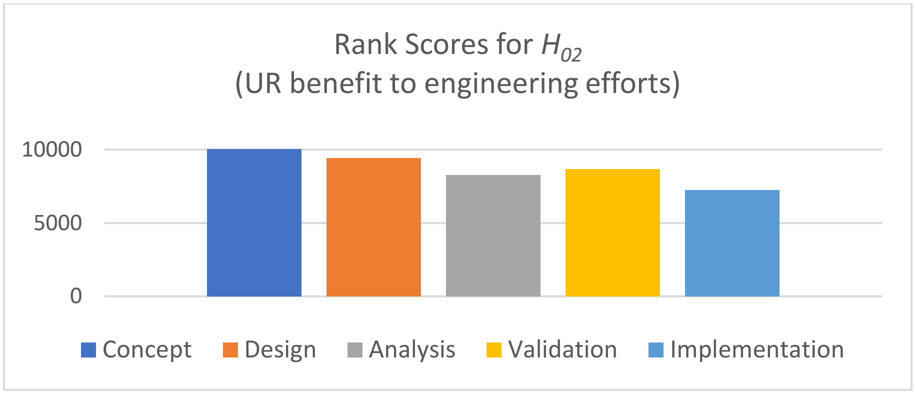


Figure 34
Kruskal-Wallis Rank Score Comparison for H_{02}

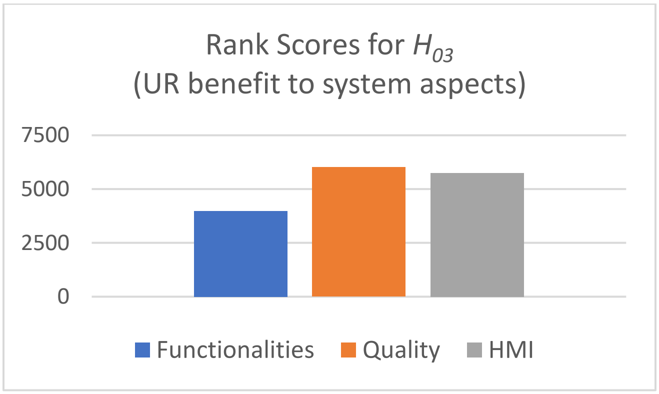


Figure 35
Kruskal-Wallis Rank Score Comparison for H_{03}

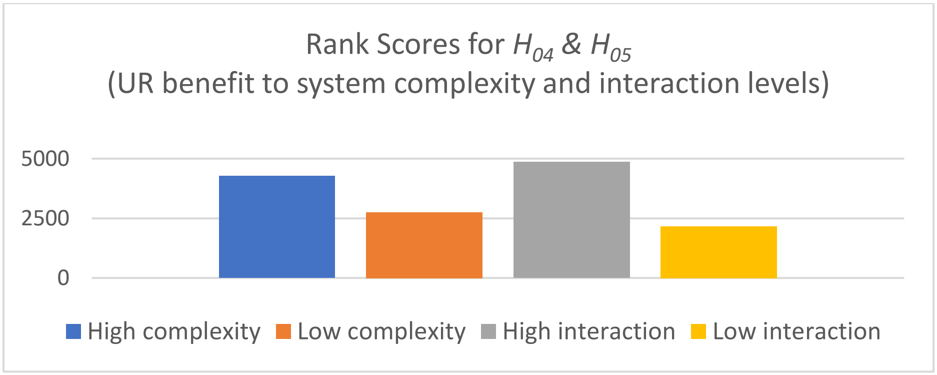


Figure 36
Kruskal-Wallis Rank Score Comparison for H_{04} & H_{05}

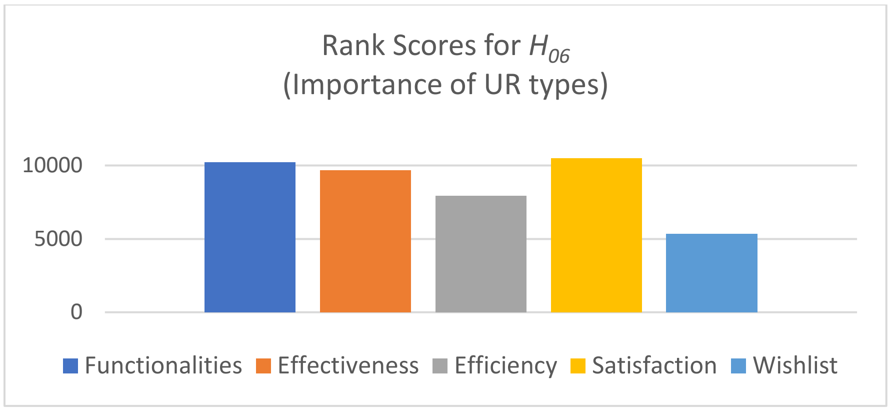


Figure 37
Kruskal-Wallis Rank Score Comparison for H_{06}

These results are descriptively summarized as follows.

- For H_{02}, although all engineering categories ranked highly (i.e. UR benefit is expected), the strongest rank was received for the system concept formation, and the lowest rank by the implementation effort.
- For H_{03}, although all system aspect categories ranked highly (i.e. UR benefit is expected), the strongest ranks were received for system quality and HMI aspects, with a lower score for functionalities.
- For H_{04} and H_{05}, systems with high complexity and/or high user interaction received significantly stronger ranks than systems with low complexity and/or low user interaction.
- For H_{06}, although all types of URs were considered important, the strongest rank was received for user satisfaction, and the lowest rank by user wishlist.
- For H_{09}, the null hypothesis is supported, and the factors cannot be considered comparably different. Each traceability aspect was considered similarly important.

5.1.3 Open Questions Feedback

The final two survey questions allowed participants to provide open-ended feedback on (a) how to improve the UR framework and/or process, and (b) general comments. Highlights are extracted and organized into the categories below. These inputs are also applied to the discussion in Chapter 6.

5.1.3.1 UR Challenges

- "Many of our customer dissatisfaction issues can be traced to not understanding our customers and use cases. To elicit URs is extremely challenging and time intensive, but it would be highly beneficial to have a robust process for doing so."
- "If done poorly, URs are dangerous as they would misguide the design. Getting the URs wrong could have sweeping consequences."
- "Writing URs can help to define a system but can be difficult to enforce. Many can seem arbitrary, and desired requirements can be ignored if thought to hinder bringing a product to market quickly."

5.1.3.2 UR Benefits

- "Shifting the terminology from the "system" to "user" will help shape engineering decisions and help make sure that the work being done is targeted at improving the customer experience."
- "For new and complex systems, gathering URs is critical, as it has a business impact."
- "The most impact from the development of URs would come from more direct customer facing systems."
- "Engineers often go off a number of assumptions, thinking that experience and knowledge will make up for actually asking the user what their requirements are. Defining URs and tracing them to the feature/system and test verification could really help solve this problem."

5.1.3.3 Process Recommendations

- "Once generic URs are written, we should establish target specifications."
- "URs should be integrated throughout the development process, so that they are considered at each process/milestone to help keep the engineering work customer focused."
- "Include URs early before creating use cases."
- "Evaluation and Acceptance Criteria should come after writing URs."

5.2 Project Results

As a separate effort from the survey research, cases studies were performed on three system development projects. Each project underwent changes after the incorporation of URs. These efforts are discussed in the sections to follow, by identifying the characteristics of each project, the methods used for analyzing the data, and the results obtained.

5.2.1 Project Characteristics

The three projects were comparatively categorized by their stage of development, system complexity, and level of user interaction. These differences provided a breath of exposure towards understanding the influence of URs to various types of projects.

Regarding their classifications, the developmental stage was based on the current project milestone, the complexity level was based on function count, and the interaction level was based on number of logical user inputs in the context diagram. The criteria for these aspects are tabulated below in Table 7.

Stage	Criteria
Early-phase	Concept formation
Mid-phase	Concept agreed
Late-phase	Implemented product

Complexity	Criteria
Low	< 5 functions
Medium	5-8 functions
High	9+ functions

Interaction	Criteria
Low	< 5 interactions
Medium	5-8 interactions
High	9+ interactions

Table 7
Criteria for Project Classifications

Each project was proprietary and therefore system-specific details are not disclosed. However, a general description is given in Table 8 below, along with the corresponding characteristics.

Project	Description	Stage	Complexity	Interaction
1	A collision warning system	Mid-phase	Medium	Medium
2	A semi-automated vehicle maneuvering system	Late-phase	High	High
3	A self-driving vehicle system	Early-phase	High	Low

Table 8
Comparison of Projects

5.2.2 Analysis Methods

The cases studies were qualitatively analyzed to evaluate the impact of inserting URs into the project efforts. To aid in this analysis and to generate insights, the number of SysML model changes were recorded for the system requirements (SRs), system functions (SFs), and validation aspects (VAs). Of interest was the percent increase in these model elements, such that a large increase would correspond to a strong influence by URs towards improving a system design and filling gaps in the specification and testing plans. Modified and removed elements were considered as well, whenever occurring.

5.2.3 System Modeling Results

The system models for each project were used as the research environment of the study, enabling the incorporation of URs along with the tracking of changes that were made as a result. These changes are broken out into the sections below.

5.2.3.1 Untraced URs

Incorporating URs into each system model led to the creation of traceability links between the URs and SRs, and between the URs and VAs. Every existing SR that served in support of one or more URs was traced using a SysML «derivedReqt» relationship, and VAs were related using the «verify» relationship. Following this effort, an average of 36% of the URs remained without traceability links, indicating the need for changes to the model. See Table 9 below.

Project	Total URs	Untraced URs
1	17	6 (35%)
2	38	8 (21%)
3	25	13 (52%)
Avg.	27	9 (36%)

Table 9
Untraced URs after Tracing to Existing SRs

The largest amount of untraced URs occurred in Project 3, possibly due to it being at the earliest stage of development. Correspondingly, the lowest number of untraced URs occurred in the late-stage Project 2.

Each untraced UR was evaluated to see if new model elements were needed. SRs and VAs were created to (a) fill documentation gaps, (b) describe design aspects which were decided to be added, and (c) capture missing test activities. As decided by the project teams, all functional changes to the system were captured as SRs and new functions. In contrast, all quality-related changes were directly captured as new VAs, without new SRs in addition.

5.2.3.2 Changes to System Requirements

Analysis revealed that of the new SRs, 48% were used to fill documentation gaps. These described functionalities which were existing or planned in the actual system, but lacked a previously written SR. While this may seem extraneous, proper documentation is a characteristic of sound engineering, in support of communication, change management, and other lifecycle activities. The remaining 52% of new SRs led to new functionalities and design aspects.

Consequently, UR incorporation led to an overall increase in requirements for each project. The new SRs represented an average increase of 15% from the initial SR set. The combined addition of URs and new SRs provided an average increase of 12% from the total requirements set. See Table 10 below.

Project	Initial Total Req's	Initial SRs	Added URs	Added SRs	SR Increase	Total Req. Increase
1	406	39	17	8	21%	6%
2	502	73	38	8	11%	9%
3	148	39	25	5	13%	20%
Avg.	352	50	27	7	15%	12%

Table 10
Changes to SRs due to UR Incorporation

5.2.3.3 Changes to System Functions

Following the new SRs, several new SFs were introduced. While the analysis of function count originally fell outside the scope of the study (as the functionalities are already captured in the SRs), it was deemed valuable to consider separately for descriptive purposes. In this context, a function is defined as a system activity with providing a customer-facing output (e.g., "Provide Steering Control", "Display HMI Overlay"). On average the UR effort led to a 10% increase in functions, shown in Table 11.

Project	Initial SFs	Added SFs	Mod. SFs	Increase
1	13	1	0	8%
2	14	2	0	14%
3	28	2	2	7%
Avg.	18	2	1	10%

Table 11
Changes to Functions due to UR Incorporation

The new functions captured the following behaviors:

- Providing interactive usage training
- Providing usage instructions
- Providing HMI status for activation
- Receiving path redirection from user

5.2.3.4 Changes to Validation Attributes

Quality-URs were compared against the existing set of VAs to establish traceability and to discover opportunities to add or modify the test plans. If a VA existed which directly validated a UR, a «verify» relationship was created between the VA's corresponding test case activities and the UR. For the remaining URs without a relation, new VAs were created to fill gaps in the validation plans. A 38% increase in VAs was observed as a result of the UR incorporation, shown in Table 12.

Project	Initial VAs	Added VAs	Increase
1	18	6	33%
2	10	5	50%
3	17	5	29%
Avg.	15	5	38%

Table 12
Changes to VAs due to UR Incorporation

Interestingly, the largest validation impact occurred to the project at the latest stage of development, as worthwhile VAs were apparently overlooked throughout a multi-year development timeline. Added VAs included the following:

- User experience aspects (overall satisfaction, feeling free, feeling safe, feeling relaxed)
- User annoyance from false events or errors
- Awareness of controls

- Awareness of user role vs. system role
- Understanding user-controlled settings
- Situational awareness during operation
- Understanding system limitations
- Physical ergonomics
- Overall system efficiency in task performance
- Changes to user behavior with system off

It can be seen there were a multitude of quality aspects added to the original validation plans for the systems. A total of 14 new aspects were captured through the use of URs, which revealed gaps in the existing documentation and thus enabled the above additions to be made.

5.2.4 Relation to Research Questions

The project case study results were included as evidence against the research questions previously stated (see section 4.3.2). Highlighted measures from the results are shown in Table 13 below.

Research Question	Case Study Evidence
Does the incorporation of URs lead to changes in the system design?	- 8% increase in design-relevant System Requirements - 10% increase in System Functions
Does the incorporation of URs lead to changes in validation testing?	- 38% increase in Validation Attributes - 14 new quality aspects captured
Does the incorporation of URs lead to unmanageability of requirements data in a project?	- Average of 27 new URs per project - 12% increase in total requirements set

Table 13
Case Study Evidence towards Research Questions

5.3 Outcome of the Study

Survey results revealed a highly positive perception on the benefits of URs within a project workflow. Most participants agreed that current methods without URs are insufficient, and a significant expectation of UR benefit was reported. Expectations were particularly strong for complex and highly interactive systems during their concept development stage. User satisfaction benefits were the most favored, and direct traceability was almost unanimously reported as being highly important.

The case study projects each showed direct benefit from the incorporation of URs into their MBSE workflow. An average of 27 URs were written for the projects, leading to an increase of system functionalities by 10%, System Requirements by 15%, and Validation Attributes by 38%.

Overall, the study results aligned with incoming expectations, supporting several advantages of URs in a project. Further discussion on the findings is provided in the following chapter.

Chapter 6. Discussion

Numerous insights can be extracted from the research findings, as URs were shown to be significantly useful in the context of real-world applications. This chapter provides a discussion on UR-related take-aways, along with an understanding of their usage context, potential pitfalls, and alternative methods. Threats to validity are also addressed via a breakdown and commentary on the study's limitations.

6.1 Interpretation and Analysis

The objective of this research effort was to apply URs in order to synthesize benefits from RE, MBSE, and HCD. The results of this study revealed an array of outcomes that align to many of the benefits provided by these fields, such as:

- Improved system functionality and quality
- Improved requirements management through traceability
- Established basis for system requirements and validation
- Expected reduction in development time, effort, and cost

These benefits align well to the research expectations that were proposed before the studies commenced, as stated in Chapter 1.4 [14, 15, 16, 17]. Further interpretation of the results is discussed in the sections below.

6.1.1 Survey Interpretations

Most survey participants conveyed a favorable impression of URs, the UR framework, and the proposed UR process. A perceived need for URs was strongly indicated, with 83% of participants reporting an insufficiency of existing methods without URs, and 96% reporting an expectation of UR-driven benefit to a project.

All types of systems and system aspects were perceived as valuable, along with all types of URs. Certain areas were considered more valuable than others, however. These front-runners included:

- Complex systems (as comparison to low-complexity)
- Systems with high user interaction (as compared to low interaction)
- Quality and HMI aspects (as compared to functionalities)
- Conceptual development (as compared to system design, analysis, implementation, or validation)
- User satisfaction aspects (as compared to efficiency, effectiveness, or wishlist aspects)

The above results lend to a recommendation to (a) incorporate URs early in a project, (b) promote UR inclusion on complex interactive systems, (c) purposefully attend to quality and HMI aspects, and (d) maintain focus on end user satisfaction throughout a project.

As a potential drawback, URs would typically increase the total informational size and complexity for a system specification. However, this consideration is balanced against the implications of missed functionalities, quality concerns, and inadequate testing efforts. The additional workload for adding URs to a system model was noted as manageable and worthwhile by 87% of participants.

6.1.2 Project Interpretations

Regarding the project case studies in which URs were added into active development efforts, certain confirmations on the expected benefits were observed. Changes were witnessed to system functional and quality aspects. Most substantially, validation testing efforts were upgraded, increasing in test coverage by 38%.

Many new system elements related to the user's understanding of the system operation. Results revealed that 50% of the added functions and 40% of the added quality criteria corresponded to this aspect. Many engineers had shared expectations that detailed user manuals would be sufficient to ensure user understanding, yet no testing was planned to evaluate this assumption. This is particularly revealing given that only an estimated 25% of users actually read user manuals for a product, and even if manuals are read they are often poorly understood [65].

The project results showed that the largest impact of URs was to the validation testing of quality aspects. The quality models discussed in Chapter 3 illustrate a multi-dimensional approach to system quality, which is highly complex and difficult to approach systematically. It can therefore be expected that this is one of the more challenging areas of systems engineering, therefore deserving of a targeted effort using a methodology such as the proposed UR framework.

Finally, an important observation can be made on the benefit of early incorporation of URs into a project. The late-phase incorporation in project 2 also experienced one of the largest impacts to SRs and VAs, leading to a degree of rework and missed opportunities in the design. Early focus on user needs enables a top-down approach to ensure objectives are met, subsystems become aligned, and that developers do not lose sight of the overall goals [66].

6.1.3 Further Insights

Additional insights were recorded by the investigator throughout the thesis study, relating to the benefits of URs:

- Non-deterministic systems are challenging to test, as the operational context is too vast to fully cover. URs can be used to focus the test scenarios towards the desired end outcomes, leading to a reduction in test cases and an increase in their relevance.
- Effective innovation is achieved by setting proper goals without constraining the technical solution early in the design process. URs work to isolate and define a set of goals for the system and serve to inspire the system requirements rather than restrict them. This enables an innovative approach and allows for multiple solutions to be compared against their abilities to meet or satisfice the URs.
- Highly complex systems contain an immense amount of interrelated information. The usage of traceability and model-based methods seems indispensable in order to manage the amount of content involved (as compared to a document-based approach).
- Successful companies such as Apple and Amazon have leveraged a strong emphasis on the customer, permeated throughout their organizations [67]. Towards this perspective, URs provide a method to represent a customer emphasis within the requirements workflow, providing a channel to cascade it throughout the engineering and stakeholder groups.
- In many cases, verification of requirements is the only type of testing performed, without formal validation testing. Although this approach ensures a system was built per its specification, it does not evaluate if a system meets user expectations [68]. Inclusion of

URs allows for a verification-style approach to validation, as the verification of URs becomes a validation effort.

6.2 Lessons Learned

Throughout the study, several lessons learned were recorded. These were taken from the open feedback received in the survey, and from comments captured during the project case studies. Selected lessons are organized by category in the sections below.

6.2.1 Lessons Learned on UR Usage

- Quality-URs can trace directly to VAs while forgoing the need for SRs. Though SRs could still be captured, they may prove challenging to write, and tend to unnecessarily repeat the content described in the UR or VA. This approach does not apply to functional-URs, for which SRs are needed to capture how a system solution serves a UR.
- When URs are plentiful in number, a prioritization method is helpful to isolate important URs (such as critical safety-related URs), or to compare URs within a trade study.
- URs need traceability to ensure they are being met and tested against. Otherwise, it is easy to ignore or forget certain content, especially across a multi-year development effort. Revisiting URs at each project milestone is advised.

6.2.2 Lessons Learned on Ontology

- Semantic confusion often exists around many of the terms and meanings used within the UR framework. Inconsistent understanding has been seen for examples such as:
    - satisfaction vs. effectiveness vs. efficiency
    - verification vs. validation vs. evaluation
    - attributes vs. criteria
    - user vs. customer
    - This lack of clarity reveals the importance of a well-defined ontology to better facilitate the communication of these topics (which are often new to many engineers).
- Applying requirement language in the form of "the user shall…" or "the user should…" is helpful for engineers to create a mental model from the user's perspective. Engineering is typically system-centric ("the system shall…"), and this user-centric sentence structure helps to readily alter entrenched perspectives.

6.2.3 Lessons Learned on Validation Testing

- The relationship between URs and acceptance criteria is particularly important to consider. In one viable approach, URs could be used to capture the basic need (e.g., "the user shall be satisfied with the operation"), while acceptance criteria could capture the specific details relevant to testing (e.g., "90% of users reporting an overall satisfaction above 80% on the post-usage survey").
- The effectiveness of validation plans is often limited by the experience and knowledge of the test engineers who create them. A more structured approach is needed for complex systems and to support collaboration across the greater team. Therefore, UR-driven testing with wide exposure to team members is recommended.

6.3 Study Limitations

The research studies contain certain threats to validity, limiting the ability to draw confident conclusions. Some prominent considerations are discussed below.

6.3.1 Participation Limitations

The sample populations lacked exposure beyond a single industry and company. Participant bias for the survey, including self-selection, was also not controlled for nor explored. The project study only involved a small number of engineering projects which were each similar in nature. The results consequently lack generalizability and should be perceived accordingly.

6.3.2 Scope Limitations

This work was built upon on model-based practices and designed primarily for interactive and complex software systems. Evaluating the research questions more broadly would require data collection and analysis across many products types, organizations, and engineering approaches. The scope of this study was substantially reduced from this idealistic approach for practical reasons.

6.3.3 Timeline Limitations

The research involved a short-term study spanning approximately one year. A more intensive study of the research questions would involve a longer-term examination of project development across the entire product lifecycle, stretching into product deployment and user feedback.

6.3.4 Process Limitations

Any benefits derived from URs can be easily degraded through their improper application. A common pitfall involves bottom-up development, in which URs are inappropriately written as a user-centric descriptions of system requirements. For example, a system requirement to "display a map graphic" could be refactored into UR form as, "The user shall receive a graphical map display". In this case, the UR merely acts as a justification of an engineering decision and could be misinterpreted as an actual user need.

Similarly, URs may be written to fulfill process guidelines without a strong concern for human-centered design, simply to "check the box" of the UR process. If so, URs are unlikely to reflect true human-centered content and thus prove ineffectual.

6.3.5 Framework Limitations

The framework itself was based on a limited number of sources and previous work, and thus does not represent a certified best practice in the art. ISO standards were an important building block, yet these are frequently revised, along with the ongoing evolution of MBSE. Additionally, and more importantly, the framework was customized around specific practices at the Case Organization in order to support the study.

6.3.6 Research Approach Limitations

It is common to attempt to advance engineering practices through the addition of structured methods (such as those proposed in this research). This assumes that work quality will consequently improve through the diligence of the participants in adhering to the new method. However, it is certainly not guaranteed that improved end outcomes will follow, and the new methods may even serve to be inadvertently detrimental via a reduction in both innovation and empowerment for the individuals involved [69].

Competing methods to URs may also provide comparable benefits. Simply by inserting an additional perspective on an engineering effort (using URs or otherwise), new findings are naturally expected. Therefore, it may not be the method which primarily delivers an impact, but the engineers themselves after being given a fresh opportunity to contribute.

6.3.7 Researcher Limitations

Certain biases are worth mentioning regarding the researcher's experience. A strong history with systems engineering (8 years) and MBSE (4 years) existed, along with experience in innovation and product design (4 years). The researcher is also under the assumption that RE, MBSE, and HCD are inherently valuable to a project, at least in comparison to other perceived alternatives.

Chapter 7. Conclusion

The research effort yielded several additional insights, discussed below. These include the implications of the findings, opportunities for other applications, and recommendations for future research.

7.1 Implications

This work creates strong case for UR inclusion within a system development project. Research results supported numerous benefits of URs: enhanced product quality, improved system functionalities, better organized human-centered information, enhanced team alignment, and improved validation testing. The inherent implication in these findings is that including URs in a project is superior than not doing so, particularly through a model-based approach.

This array of upsides and the powerful nature of these advantages is quite promising for a wide variety of applications. However, these notions should be balanced against a consideration for potential drawbacks of including URs, such as:

- Added complexity in the engineering process and tool
- Difficulties in generating well-formulated URs, particularly for quality attributes
- Challenges related to MBSE in general [70]
- Over-constraining an innovative workflow

It is hoped that this work provides an encouraging perspective for the improvement of human-centered requirements development. Adopting this knowledge is ultimately expected to enhance the methods applied in developing a new product, and consequently improving the end-product itself.

7.2 Additional Applications

This research was mainly focused on user-interactive software systems. However, the methods could be applied to other systems as well. The UR framework and process are not strictly formulated and could be easily adapted to suit the needs of other efforts. These may include:

- **Physical or cyber-physical products**. These systems may include ergonomic considerations which are not shown in the UR quality model.
- **Process development**. These efforts still involve "users" but not in the traditional product usage sense. A modified UR framework could maintain a model-based approach using the Business Process Model and Notation [71].
- **Enterprise architecture**. Organizational modeling can be supporting using methods such as TOGAF [72], which places requirements as central in the framework.

An adaptation of the UR methods presented in this research is expected per the needs of an application. Certain principles are recommended to be maintained during modification:

- Isolating discrete human-centered aspects and achieving traceability to their related informational elements
- Using human-centered content as the basis for design and validation testing
- Using a quality model to better discover an array of quality concerns and managing these separately from functional content

7.3 Recommendations for Future Research

Future research on this topic would provide useful insights to compliment those provided here. Given that the work was performed at a single Case Organization within a narrow timeframe, additional studies would serve to validate and extend the findings in a valuable manner.

Certain extensions of the research were left open for future focus. These included:

- Techniques for analyzing URs, particularly regarding their coverage of all relevant and important system aspects
- Establishing a method for capturing the variety of human-centered content before authoring URs
- Investigating model-based methods to replace requirements with other model elements in the system specification (e.g., use cases or activities to replace functional-URs, and acceptance criteria to replace quality-URs [73])

The relationship between URs and validation testing is also noted as a reasonable area for further work. Topics could include:

- Identifying how to incorporate acceptance criteria more clearly in the UR framework
- Investigating a validation-driven approach to system development in place of a requirements-driven approach
- Establishing a more formal model-based method for specifying validation tests

Regarding UR application, other work could be considered, including:

- Experimenting with the UR approach in different contexts (i.e. different companies, industries, project methodologies, product types, etc.)
- Investigating the impact of URs in non-model-based projects
- Re-evaluating the approach in a repeat fashion

Finally, the UR framework and process described in this work represents one particular approach to UR inclusion. Research to present other UR and UR-related methods would be extremely valuable, as this research presents as viable approach as opposed to a sole best technique.

7.4 Conclusion

In summary, this research explored the inclusion of URs in a system development project, encompassing their potential, expected, and observed benefits. A proposed UR method was provided, including a novel model-based framework and a suitable process for its incorporation into a variety of applications. The value of the method was evaluated in a real-world setting, revealing its ability to enhance and supplement the engineering of a new complex system. Looking forward, interested parties may consider leveraging this work by pragmatically adapting it to their existing practices.

References

[1] M. Richter and M. Fluckiger, *User-Centred Engineering*, Heidelberg: Springer, 2014.

[2] J. A. Bargas-Avila and K. Hornbæk, "Old wine in new bottles or novel challenges: a critical analysis of empirical studies of user experience," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, New York, NY, 2011.

[3] A. Sutcliffe, *User-Centred Requirements Engineering*, London: Springer, 2002.

[4] T. Bell and T. Thayer, "Software requirements: Are they really a problem?," in *Proceedings 2nd International Conference on Software Engineering*, Los Alamitos, CA, 1976.

[5] Standish Group, "Chaos Report," http://www.standishgroup.com, 1994-2014.

[6] K. Eason, *Information Technology and Organisational Change*, London: Taylor and Francis, 1988.

[7] M. Tseng and J. Jiao, "Computer-aided requirements management for product definition: A methodology and implementation.," *Concurrent Engineering: Research and Applications,* vol. 6, no. 2, pp. 145-160, 1998.

[8] L. Almefelt, F. Berglund, P. Nilsson and J. Malmqvist, "Requirements management in practice: Findings from an empirical study in the automotive industry.," *Research in Engineering Design,* vol. 17, pp. 113-134, 2006.

[9] B. Ramesh, C. Stubbs, T. Powers and M. Edwards, "Requirements Traceability: Theory and Practice," *Annals of Software Engineering,* vol. 3, no. 1, pp. 397-415, 1997.

[10] B. H. Cheng and J. M. Atlee, "Research Directions in Requirements Engineering," in *Future of Software Engineering*, Minneapolis, MN, 2007.

[11] J. Fernandes, E. Henriques and A. Silva, "Requirements change in complex technical systems: an empirical study of root causes.," *Res Eng Design,* vol. 26, pp. 37-55, 2015.

[12] B. Boehm, "Requirements that handle IKIWISI, COTS, and rapid change," *Computer,* vol. 33, no. 7, pp. 99-102, 2000.

[13] J. Satzinger, R. Jackson and S. D. Burd, *Systems Analysis and Design in a Changing World*, 7 ed., Boston: Course Technology, 2015.

[14] NIST, "Common Industry Specification for Usability - Requirements," Information Access Division, Information Technology Laboratory, National Institute of Standards and Technology, 2007.

[15] INCOSE, "Systems Engineering Handbook A Guide for System Life Cycle," International Council on Systems Engineering, San Diego, CA, 2011.

[16] N. Bevan, J. Carter, J. Earthy, T. Geis and S. Harker, "What are user requirements? Developing an ISO standard.," in *Human-Computer Interaction. Theories, Methods, and Human Issues. 20th International Conference, HCI International.*, Las Vegas, NV, 2018.

[17] T. E. White, "Assessing the Impact of Requirements Review on Quality Outcomes.," PhD Diss., George Washington University, 2018.

[18] M. I. Bugaje, "A novel framework for user-centered research data," PhD Diss., Northumbria University, 2019.

[19] M. Maguire and N. Bevan, "User Requirements Analysis: A Review of Supporting Methods," in *IFIP World Computer Congress*, Deventer, Netherlands, 2002.

[20] M. Cohn, *User Stories Applied: For Agile Software Development*, Boston: Addison-Wesley, 2004.

[21] N. A. Stanton, "Hierarchical task analysis: Developments, applications, and extensions," *Applied Ergonomics,* vol. 37, no. 1, pp. 55-79, 2006.

[22] U.S. Air Force, *SMC Systems Engineering Primer & Handbook*, 2nd ed., 2004.

[23] DoD Systems Management College, *Systems Engineering Fundamentals*, Fort Belvoir, VA: Defense Acquisition University Press, 2001.

[24] BKCASE, *Systems Engineering Body of Knowledge (SEBoK),* INCOSE, 2020.

[25] M. Glinz, "On Non-Functional Requirements," in *15th IEEE International Requirements Engineering Conference*, Delhi, India, 2007.

[26] K. Pohl and C. Rupp, *Requirements Engineering Fundamentals*, Berlin: Springer-Verlag, 2010.

[27] D. Ameller, C. Ayala, J. Cabot and X. Franch, "Non-Functional Requirements in Architectural Decision-Making," *IEEE Software,* vol. 30, no. 2, pp. 61-67, 2013.

[28] L. Chung, B. Nixon, E. Yu and J. Mylopoulos, *Non-functional Requirements in Software Engineering*, Dordrecht: Kluwer Academic, 2000.

[29] B. Lawrence, K. Wiegers and C. Ebert, "The top ten risks of requirements engineering," *IEEE Software,* vol. 18, no. 6, pp. 62-63, 2001.

[30] J. Mylopoulos, L. Chung and E. Yu, "From Object-Oriented to Goal-Oriented Requirements Analysis," *Communications of the ACM,* vol. 42, no. 1, pp. 31-37, 1999.

[31] K. Brennan, A Guide to the Business Analysis Body of Knowledge Version 2.0, Pickering, ON: International Institute of Business Analysis, 2009.

[32] ISO/IEC 15288, "Systems and software engineering — System life cycle processes," 2015.

[33] ISO/IEC/IEEE 29148, "Systems and software engineering — Life cycle processes — requirements engineering," 2018.

[34] ISO 25065, "Systems and software engineering — Software product Quality Requirements and Evaluation (SQuaRE) — Common Industry Format (CIF) for Usability: User requirements specification," 2019.

[35] J. A. McCall, P. K. Richards and G. F. Walters, "Factors in Software Quality," Rome Air Development Center, Griffiss Air Force Base, New York, 1977.

[36] B. W. Boehm, J. R. Brown, H. Kaspar, M. Lipow, G. McLeod and M. Merritt, *Characteristics of Software Quality*, Amsterdam: North Holland, 1978.

[37] T. P. Bowen et. al., "Specification of software quality attributes," Rome Air Development Center. Rep. RADC-TR-85-37, Griffiss Air Force Base, NY, 1985.

[38] J. Nielsen, *Usability Engineering*, Cambridge: AP Professional, 1993.

[39] ISO/IEC 25010, "Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models," 2011.

[40] J. Bøegh, "A New Standard for Quality Requirements," *IEEE Software,* pp. 57-63, 2008.

[41] B. Jones and M. Ryan, "ISO/IEC 15288(E)—Visual Enhancement for Effective Communication," in *Systems Engineering / Test and Evaluation Conference*, Canberra, Australia, 2011.

[42] ISO/IEC 12207, "Systems and software engineering — Software life cycle processes," 2017.

[43] ISO 9241-210, "Ergonomics of human-system interaction — Part 210: Human-centred design for interactive systems," 2019.

[44] ANSI/EIA 632, "Processes for Engineering a System," American National Standards Institute (ANSI)/Electronic Industries Association (EIA), Philadelphia, PA, 2003.

[45] K. Beck, J. Grenning, R. C. Martin, M. Beedle, J. Highsmith, S. Mellor, A. v. Bennekum, A. Hunt, K. Schwaber, A. Cockburn, R. Jeffries, J. Sutherland, W. Cunningham, J. Kern,

D. Thomas, M. Fowler and B. Marick, "Manifesto for Agile Software Development," Agile Alliance, 2001.

[46] K. Schwaber, *Agile Project Management with Scrum*, London: Pearson Education, 2004.

[47] D. Anderson, *Kanban: Successful Evolutionary Change for Your Technology*, Washington DC.: Blue Hole Press, 2010.

[48] K. Beck, *Extreme Programming Explained: Embrace Change*, Boston: Addison-Wesley, 2000.

[49] E.-M. Schön, J. Thomaschewski, M. Mejías and M. J. Escalona, "A Metamodel for Agile Requirements Engineering," *Journal of Computer and Communications,* vol. 7, pp. 1-22, 2019.

[50] S. Friedenthal, A. Moore and R. Steiner, *Practical Guide to SysML: Systems Modeling Language*, Waltham, MA: Morgan Kaufmann Publishers Inc., 2008.

[51] OMG, "SysML v1.6," Object Management Group, 2019.

[52] OMG, "UML v2.5.1," Object Management Group, 2017.

[53] K. Henderson and A. Salado, "Value and benefits of model-based systems engineering (MBSE): Evidence from the literature," *INCOSE Systems Engineering,* vol. 24, no. 1, pp. 51-66, 2021.

[54] R. Ali, F. Dalpiaz and P. Giorgini., "A Goal-based Framework for Contextual Requirements Modeling and Analysis," *Requirements Engineering,* vol. 15, no. 4, pp. 439-458, 2010.

[55] A. Cockburn, *Writing Effective Use Cases*, New Delhi: Dorling Kindersley Pvt Ltd, 2000.

[56] A. A. Anda, "Modeling Adaptive Socio-Cyber-Physical Systems with Goals and SysML," in *IEEE 26th International Requirements Engineering Conference*, Banff, AB, Canada, 2018.

[57] ITU-T, "Z.150 User Requirements Notation (URN)," International Telecommunication Union, 2011.

[58] D. Amyot, A. A. Anda, M. Baslyman, L. Lessard and J.-M. Bruel, "Towards Improved Requirements Engineering with SysML and the User Requirements Notation," in *IEEE 24th International Requirements Engineering Conference*, Beijing, China, 2016.

[59] Federal Ministry for Economic Affairs and Energy, "PEGASUS Research Project," 2021. [Online]. Available: https://www.pegasusprojekt.de/en/. [Accessed 01 2021].

[60] E.-M. Schön, J. Thomaschewski and M. J. Escalona, "Agile Requirements Engineering: A systematic literature review," *Computer Standards & Interfaces,* vol. 49, pp. 79-91, 2017.

[61] A. Bowling, "Data collection methods in quantitative research: questionnaires, interviews," in *Research Methods in Health: Investigating Health and Health Services*, Buckingham, Open University Press, 1997, pp. 257-272.

[62] G. Sullivan and A. R. Artino, "Analyzing and Interpreting Data From Likert-Type Scales," *Journal of Graduate Medical Education,* vol. 5, no. 4, pp. 541-542, 2013.

[63] N. B. Robbins and R. M. Heiberger, "Plotting Likert and Other Rating Scales," in *JSM Proceedings, Section on Survey Research Methods*, Boston, American Statistical Association, JSM 2011, p. 1058–1066.

[64] D. L. Clason and T. J. Dormody, "Analyzing Data Measured by Individual Likert-Type Items," *Journal of Agricultural Education,* vol. 35, no. 4, pp. 31-35, 1994.

[65] A. L. Blackler, R. Gomez, V. Popovic and M. H. Thompson, "Life Is Too Short to RTFM: How Users Relate to Documentation and Excess Features in Consumer Products," *Interacting with Computers,* vol. 28, no. 1, p. 27–46, 2016.

[66] P. Oppenheimer, *Top-Down Network Design*, 3rd ed., Indianapolis: Cisco Press, 2011.

[67] J. Bean and S. V. Tyne, *The Customer Experience Revolution: How Companies Like Apple, Amazon, and Starbucks Have Changed Business Forever*, St. Johnsbury: Raphel Marketing, Incorporated, 2011.

[68] NASA, *NASA Systems Engineering Handbook v2*, National Aeronautics and Space Administration, 2016.

[69] G. Hamel and M. Zanini, *Humanocracy: Creating Organizations as Amazing as the People Inside Them*, Boston: Harvard Business Press, 2020.

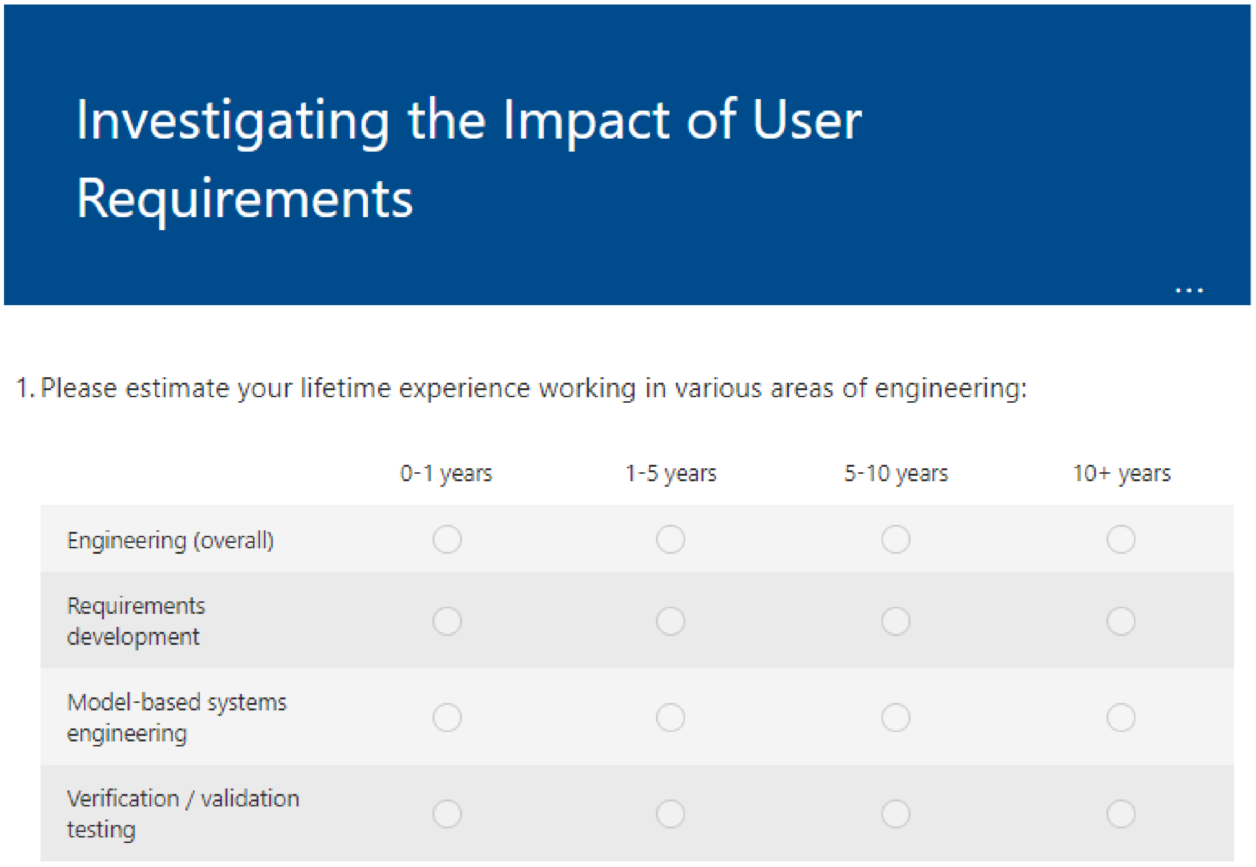
[70] l. Albers and C. Zingel, "Challenges of Model-Based Systems Engineering: A Study towards Unified Term Understanding and the State of Usage of SysML," *Smart Product Engineering. Lecture Notes in Production Engineering,* pp. 83-92, 2013.

[71] OMG, "Business Process Model and Notation v2.0," Object Management Group, 2010.

[72] The Open Group, "The Open Group Architecture Framework (TOGAF) Standard v9.2," 2018.

[73] K. Pugh, *Lean-Agile Acceptance Test-Driven Development: Better Software Through Collaboration*, Boston: Addison-Wesley, 2011.

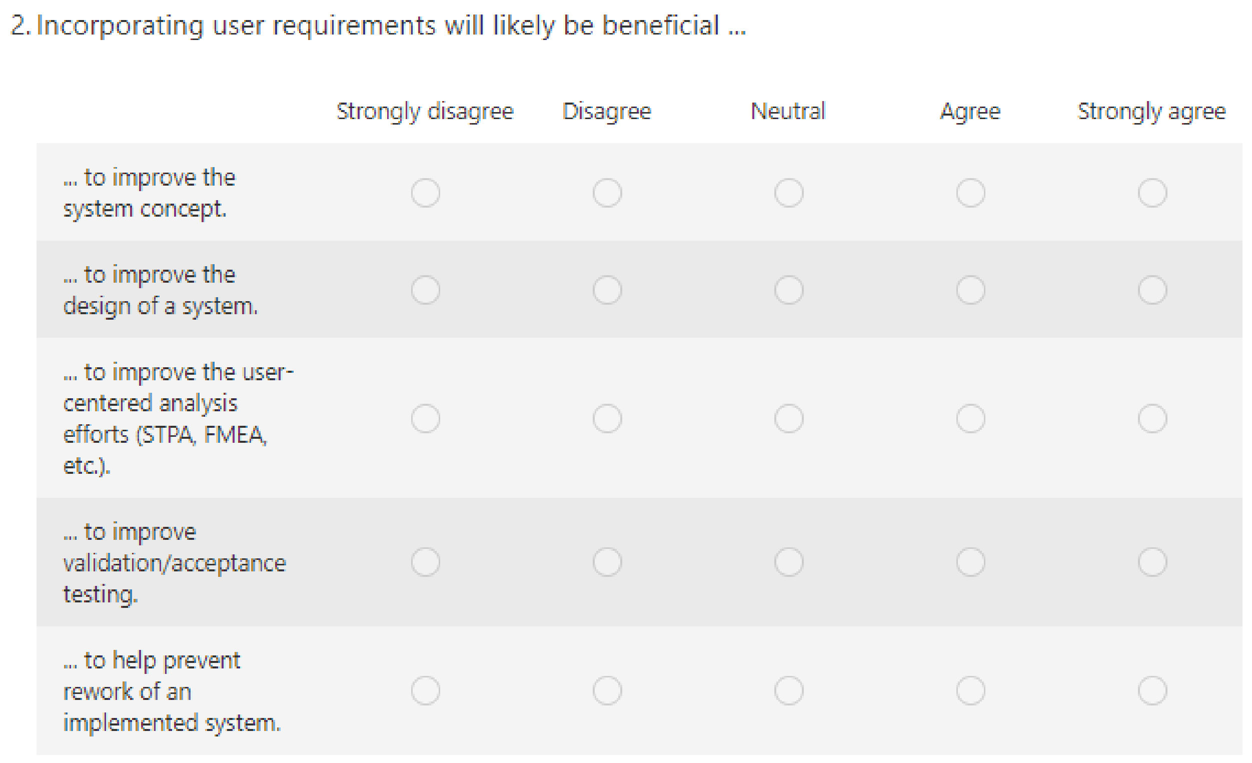
Appendix: Survey Conducted



Investigating the Impact of User Requirements

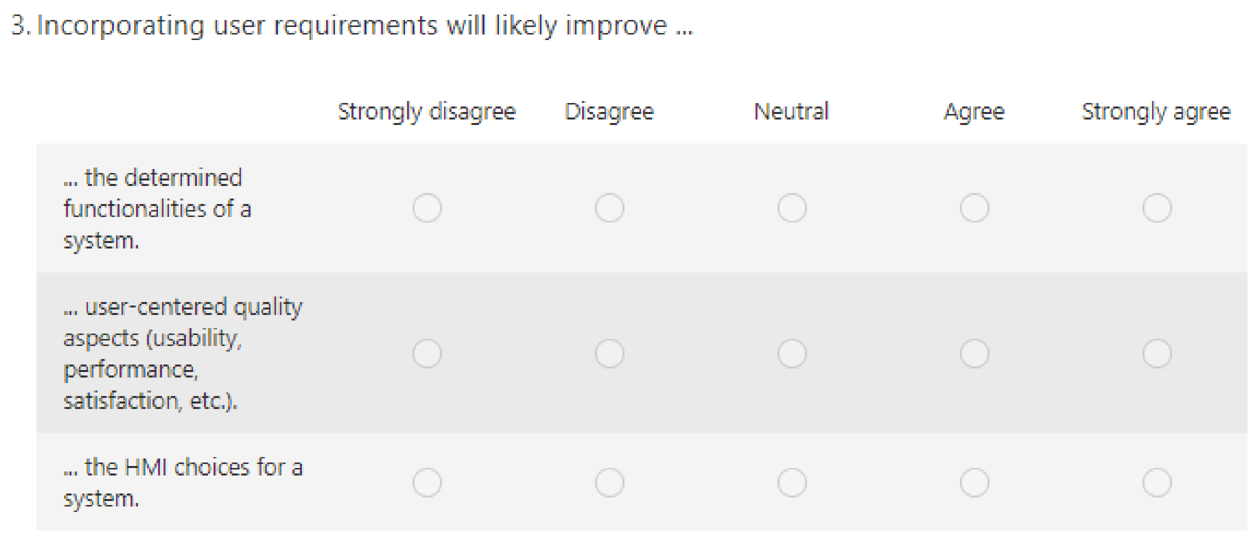
1. Please estimate your lifetime experience working in various areas of engineering:

	0-1 years	1-5 years	5-10 years	10+ years
Engineering (overall)	○	○	○	○
Requirements development	○	○	○	○
Model-based systems engineering	○	○	○	○
Verification / validation testing	○	○	○	○



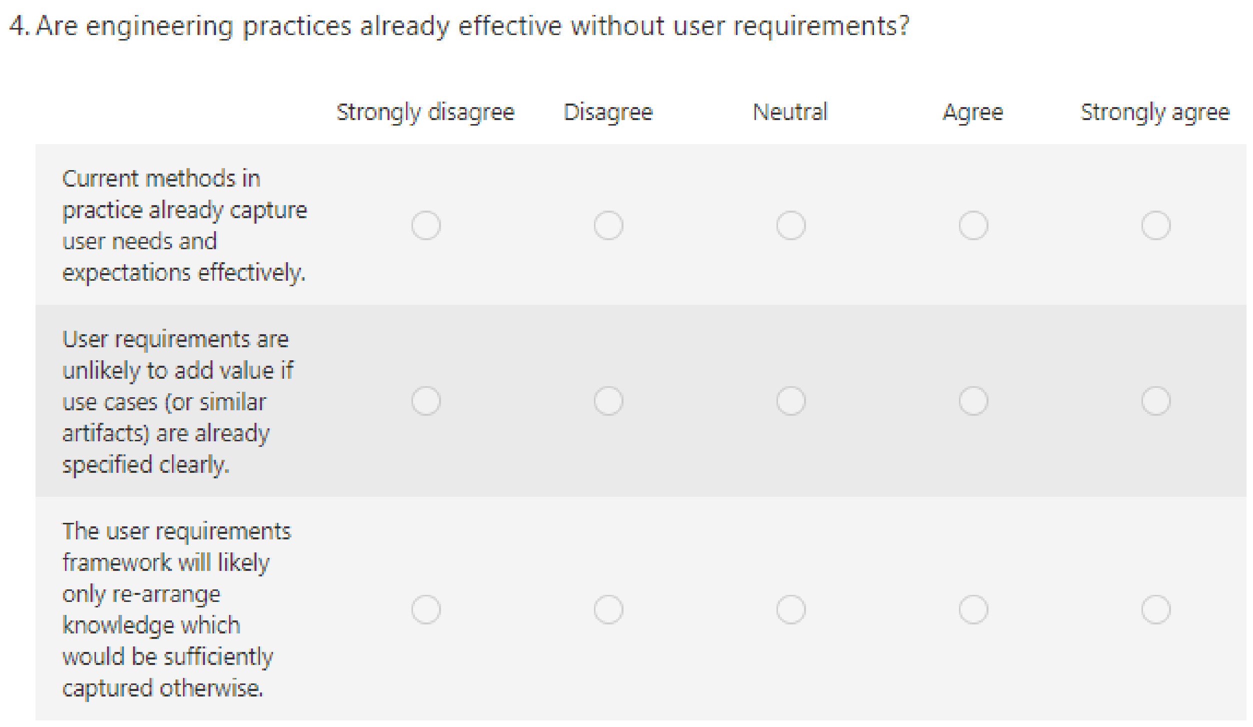
2. Incorporating user requirements will likely be beneficial ...

	Strongly disagree	Disagree	Neutral	Agree	Strongly agree
... to improve the system concept.	○	○	○	○	○
... to improve the design of a system.	○	○	○	○	○
... to improve the user-centered analysis efforts (STPA, FMEA, etc.).	○	○	○	○	○
... to improve validation/acceptance testing.	○	○	○	○	○
... to help prevent rework of an implemented system.	○	○	○	○	○



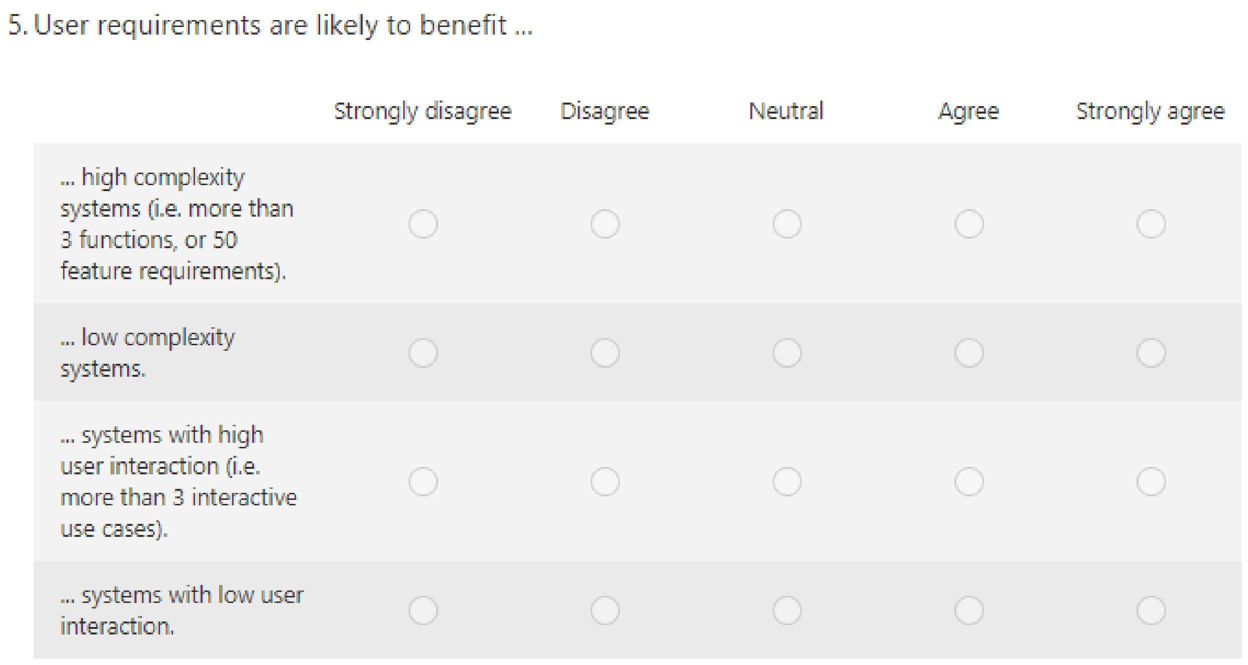
3. Incorporating user requirements will likely improve ...

	Strongly disagree	Disagree	Neutral	Agree	Strongly agree
... the determined functionalities of a system.	○	○	○	○	○
... user-centered quality aspects (usability, performance, satisfaction, etc.).	○	○	○	○	○
... the HMI choices for a system.	○	○	○	○	○



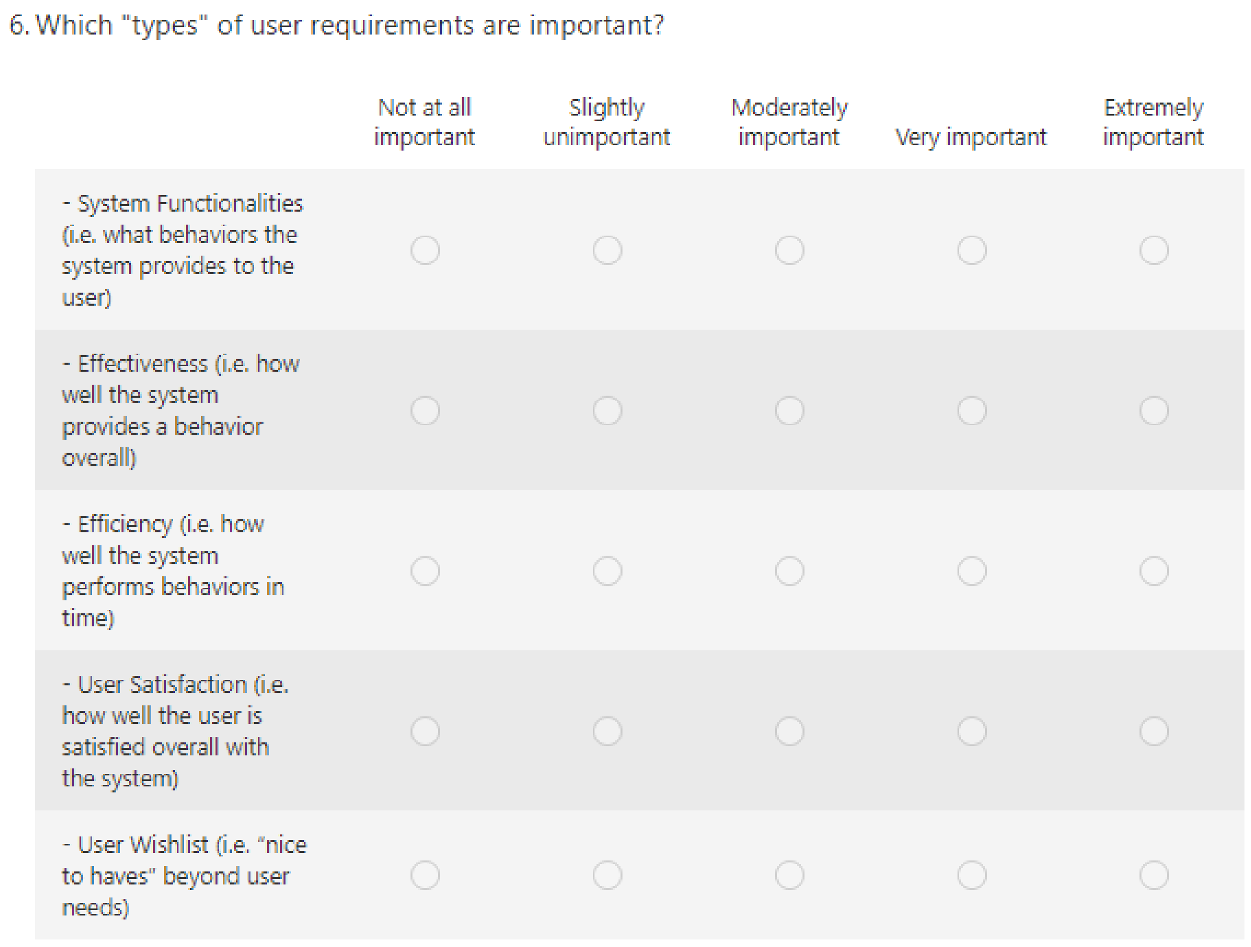
4. Are engineering practices already effective without user requirements?

	Strongly disagree	Disagree	Neutral	Agree	Strongly agree
Current methods in practice already capture user needs and expectations effectively.	○	○	○	○	○
User requirements are unlikely to add value if use cases (or similar artifacts) are already specified clearly.	○	○	○	○	○
The user requirements framework will likely only re-arrange knowledge which would be sufficiently captured otherwise.	○	○	○	○	○



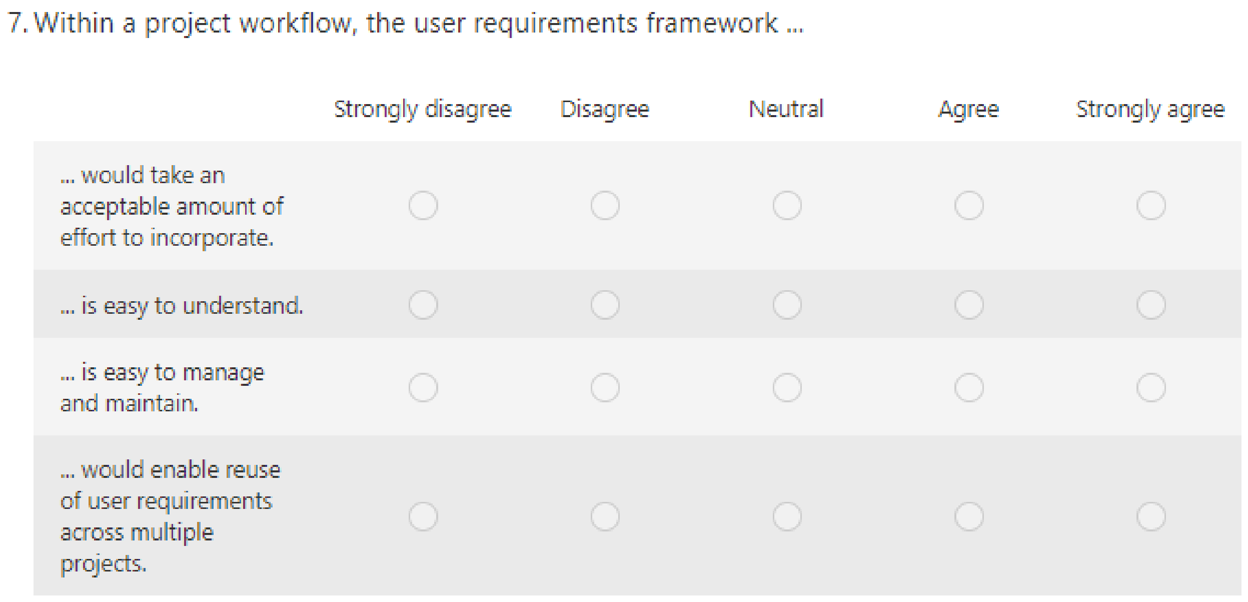
5. User requirements are likely to benefit ...

	Strongly disagree	Disagree	Neutral	Agree	Strongly agree
... high complexity systems (i.e. more than 3 functions, or 50 feature requirements).	○	○	○	○	○
... low complexity systems.	○	○	○	○	○
... systems with high user interaction (i.e. more than 3 interactive use cases).	○	○	○	○	○
... systems with low user interaction.	○	○	○	○	○



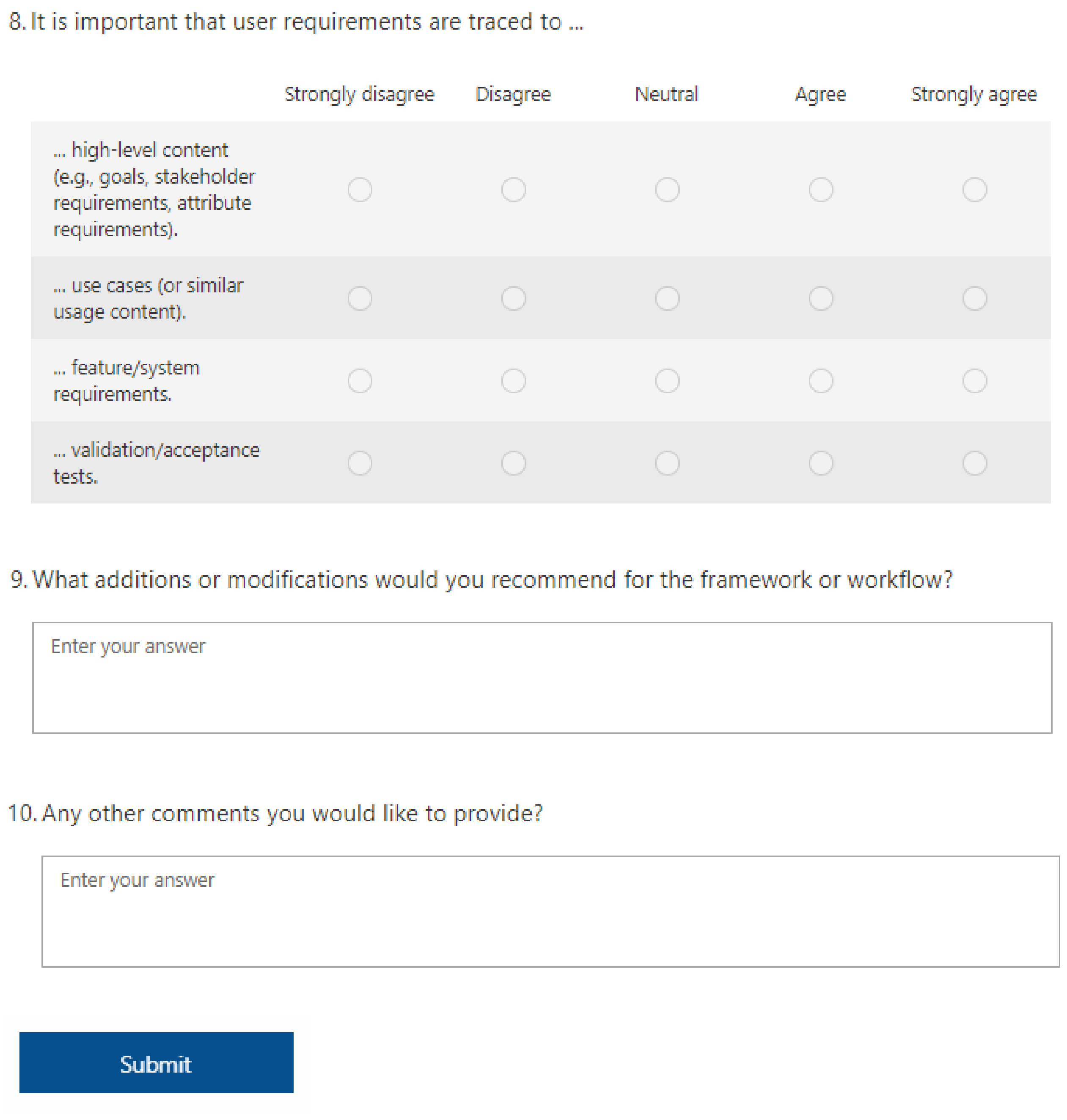
6. Which "types" of user requirements are important?

	Not at all important	Slightly unimportant	Moderately important	Very important	Extremely important
- System Functionalities (i.e. what behaviors the system provides to the user)	○	○	○	○	○
- Effectiveness (i.e. how well the system provides a behavior overall)	○	○	○	○	○
- Efficiency (i.e. how well the system performs behaviors in time)	○	○	○	○	○
- User Satisfaction (i.e. how well the user is satisfied overall with the system)	○	○	○	○	○
- User Wishlist (i.e. "nice to haves" beyond user needs)	○	○	○	○	○



7. Within a project workflow, the user requirements framework ...

	Strongly disagree	Disagree	Neutral	Agree	Strongly agree
... would take an acceptable amount of effort to incorporate.	○	○	○	○	○
... is easy to understand.	○	○	○	○	○
... is easy to manage and maintain.	○	○	○	○	○
... would enable reuse of user requirements across multiple projects.	○	○	○	○	○



8. It is important that user requirements are traced to ...

	Strongly disagree	Disagree	Neutral	Agree	Strongly agree
... high-level content (e.g., goals, stakeholder requirements, attribute requirements).	○	○	○	○	○
... use cases (or similar usage content).	○	○	○	○	○
... feature/system requirements.	○	○	○	○	○
... validation/acceptance tests.	○	○	○	○	○

9. What additions or modifications would you recommend for the framework or workflow?

Enter your answer

10. Any other comments you would like to provide?

Enter your answer

Submit