Capable - A Framework for Assessing Software Architecture Development Capability for Public Health Information Systems Projects

Anthony Dang
University of Technology Sydney, anthony.j.dang@student.uts.edu.au

Ghassan Beydoun
University of Technology Sydney, Ghassan.Beydoun@uts.edu.au

Capable - A Framework for Assessing Software Architecture Development Capability for Public Health Information Systems Projects

Anthony Dang
School of Computer Science - Faculty of Engineering and Information Technology
University of Technology Sydney
Australia
Email: Anthony.J.Dang@student.uts.edu.au

Ghassan Beydoun
School of Computer Science - Faculty of Engineering and Information Technology
University of Technology Sydney
Australia
Email: Ghassan.Beydoun@uts.edu.au

Abstract

Context: In the public health domain, the prevalence of unsuccessful Information Systems projects is notable. Technical issues persist beyond the pervasive problems of inflated budgets and extended deadlines. These include poor usability, system instability, suboptimal performance, and data inconsistencies. These undesirable outcomes are linked to the Software Engineering process and the Software Architecture underpinning the system. To mitigate these issues, a project's capability to achieve Software Architecture quality must be assessed. The socio-technical nature of Information Systems projects in the Public Health Domain necessitates a holistic approach.

Aim: To address the need to assess Software Architecture development capability within the context of the Public Health Information Systems.

Method: The framework was synthesised and evaluated using Design Science Research. The synthesis incorporated Australian and American Public Health Information Systems failure exemplars and drew upon the existing Software Engineering literature. The framework's theoretical constructs were evaluated using an unstructured data source (government audit reports of Public Health Information Systems failures).

Results: The conceptual aspects of the framework were evaluated. The framework was capable of detecting failure scenarios. Furthermore, the framework provided an indicative capability score and capability grade while providing possible actions for improving the project's Software Architecture development capability.

Keywords: Public Health, Information Systems, Success Factors, Failure Factors, Software Architecture, Software Engineering, Capability, Assessment, Design Science Research.

1 Introduction

Public Health Information Systems (PHIS) projects have notably suffered from poor end-user outcomes. These systems may manifest issues related to poor usability, system instability, poor performance, and data inconsistencies. The pervasiveness of these Information System (IS) project failures has led some to believe these failures are "part of modern life" (Ludlow 2016). A notable exemplar in Australia was the Queensland health sector payroll system (Eden and Sedera 2014). At $1.25 billion (AUD), the software problems resulted in staff being either underpaid, overpaid, or even not paid at all.

Underpinning software quality is the Software Architecture (SA). We posit that given the human impact of PHIS project failures, *it is important to determine a project's capability to achieve successful SA*. A project's SA development capability would benefit from being assessed before the project commences and reassessed during critical stages of the development lifecycle. This research aims to address the need for assessing the capability of developing SA within PHIS projects. Therefore, we posit the Research Question: *Is it possible to assess a project's capability to achieve successful SA?* To the best of our knowledge, this appears to be a gap in the literature, which we address in this paper.

In our previous paper (published at ACIS 2023), we asserted that a lack of appropriate SA underpins many of the issues related to poor end-user outcomes (Dang and Beydoun 2023). We found that SA quality is not simply a concern for the software vendor. It concerns all parties contributing to the project, including the government organisations. This multi-disciplinary research represents a continuation of our previous work, incorporating Software Engineering (SE), SA, and PHIS. Our holistic approach to capability assessment brings to bear the social and organisational impacts on the software architecting process (Galster et al. 2017). In this paper, we present:

Capable - An empirically validated framework for assessing Software Architecture development capability for Public Health Information Systems Projects

The intended Framework Users include individuals in technical roles, such as Chief Technical Officers, Technical Project Managers, Software Architects, and technical individuals within government organisations who seek to initiate and continually assess PHIS projects. To the best of our knowledge, such a framework for assessing the SA development capability of PHIS projects is a novel contribution.

The framework was developed using Design Science Research methodology (DSR) and evaluated using documented exemplars of PHIS failures.

2 Background and Related Work

The success and failure of IS projects have been studied for some time. Success may be defined in terms of project sponsor satisfaction, end-user satisfaction, stakeholder satisfaction, and meeting technical specifications (Silvius and Schipper 2015). Factors impacting success include project planning, corporate culture, project management, user involvement, and communication (Yeo 2002). Other factors may include information quality, individuals, and organisations (Li 1997; McManus and Wood-Harper 2007; Yeo 2002). In one study, factors were found to include: trust, user expectations, motivation, IT infrastructure, relationship with developers, domain knowledge, management support, management processes, and organisational competence (Petter et al. 2013). Several evaluation models for IS success exist (Varajão et al. 2022). However, the models do not focus on SA or assess a project's capability to develop the SA.

One factor that may impact success is the instability of government organisations due to political changes (Ombudsman 2024). These changes may occur with little notice, reflecting shifts in policy priorities with the election of new governments. Such changes can impact ongoing projects, including IT initiatives, leading to delays, budget overruns, and the need for re-scoping. A New Zealand study showed that the government organisations' main sponsor's accountability and inclination toward Agile Project Management methodology were key to a project's success (Douglas 2021).

Successful software development has been discussed in the context of Technical Debt - A metaphor commonly used in SE to denote implementation deficiencies that can lead to problems with large scope. Architectural Technical Debt is a related concept (Verdecchia et al. 2021). In one study, it was found that architectural design decisions may incur Architectural Technical Debt, as well as Requirements Debt (the delta between the specified requirements and the implemented system) (Soliman et al. 2021). The researchers stated that knowledge capturing of debt-incurring decisions could improve outcomes. A theory of Architectural Technical Debt has been proposed (Verdecchia et al. 2021). This included causes relating to knowledge, human factors, complex business processes, unsuitable architectural decisions, incorrect implementation of correct architecture, and lack of anticipation of the potential evolution of the architecture. The causal factors are shown in Appendix 2. This is a holistic model of factors that can impact SA. Causal categories that have been previously identified include planning and management, knowledge, methodology/practices, organisation, people and documentation (Rios et al. 2018). Acknowledging these causal categories is necessary for understanding how to approach addressing capability.

The Capability Maturity Model Integration (CMMI) provides a performance improvement framework designed to assess and enhance the maturity of an organisation's processes (Institute 2024). Encompassing a range of disciplines, CMMI outlines best practices for requirements management and defect prevention. Complementing this, ISO 9001 specifies a quality system for software development. It provides a framework for organisations to assess their ability to deliver quality products and services (ISO 2015). While CMMI and ISO 9001 have necessary aspects, they do not address SA development capability. SA competence is multifaceted (Bass et al. 2021), encompassing individual, team and organisational capabilities. For individual architects, architecture competence necessitates a robust skillset that blends technical knowledge of programming languages and design patterns with effective project management practices and strong communication abilities (Bass et al. 2021). At the organisational level, architecture competence necessitates identifying and prioritising architectural requirements, ensuring the architecture evolves throughout the project lifecycle. In addition, Bass et al. discuss the need for architects and organisations to embrace agile methodologies while maintaining architectural rigour (Bass et al. 2021). While these competence aspects have been named, a comprehensive assessment of capability for SA in PHIS projects remains unaddressed.

Our previous work (published at ACIS2023) asserted the need for a framework to determine SA success in PHIS projects (Dang and Beydoun 2023). We derived our model from the Unified Model of Information System Development Success (Siau et al. 2010), which incorporates various inputs and processes. The input categories in our original model include Project Characteristics, Organisation, Requirements, Team, Knowledge & Skills, and Human Factors. The process categories include Project Management, Communication, Practices, Techniques, and Decision-making. These high-level categories are depicted in Appendix 1.

In the following sections, we outline our framework's research methodology, synthesis, and evaluation.

3 Assessing Software Architecture Development Capability

For some time, it has been understood that success requires an appropriate configuration of organisational, behavioural, cognitive, and social factors (Kaplan and Harris-Salamone 2009). Indeed, the SA process and architectural decision-making have been shown to be impacted by factors such as cost, risk, requirements, tools, and business goals (Demir et al. 2024). It follows that *the primary goal of this research is to assess the capability of a project to achieve successful SA.* To this end, the Design Science Research (DSR) methodology (Vaishnavi and Kuechler 2004) was selected to conduct our research. DSR was chosen as it necessitates the development of innovative solutions to real-world problems. It enables socio-technical solutions to be developed by creating and evaluating artifacts that improve knowledge and practice (Gregor and Hevner 2013). DSR methodology facilitated the iterative development and refinement of an artifact derived from theoretical constructs in the literature. This section outlines the principles for constructing a framework for assessing the SA development capability of a PHIS project instance.

3.1 Design Principles for the Capability Assessment Instrument

When the design of a socio-technical artifact is complex (in terms of its size and the number of components), then explicit extraction of design principles may be required (Gregor and Hevner 2013). The framework incorporates many factors encompassing the socio-technical categories shown in Appendix 1. We anticipate that our framework will be utilised at various stages of the project lifecycle. For example, in the requirements phase, the framework can help assess whether the PHIS project has a low capability, allowing the Framework Users to address any shortcomings before proceeding. As the project progresses, changing factors might necessitate reapplying the framework, which could influence decisions to continue or halt the project to address the shortcomings. This is particularly crucial if requirements evolve or new ones are introduced. Therefore, the framework should be capable of assessment before and during the project's lifecycle. In addition, for use in practice, the framework must have a high degree of usability *(Nielsen 1994)*. The guiding design principles for creating the framework are:

Comprehensiveness - The framework should perform a holistic socio-technical assessment of the project, considering the breadth of factors that can impact SA outcomes.

Simplicity and Clarity - The framework should be formulated in clear language to ensure that Framework Users can easily understand and accurately interpret the items. This reduces the likelihood of misinterpretation and enhances the reliability of the Framework Users' application of the framework.

Reliability – The framework should be based on well-established theories and empirical research. It should deliver assessments for small, medium, and large-scale projects, providing consistent performance.

User-Centred Design - The framework should be intuitive and easy to use, allowing easy data input and result interpretation without extensive training. It should guide Framework Users step by step through the assessment process.

Actionability - The assessment results should be actionable, providing insights and recommendations for improving socio-technical factors in the PHIS project.

3.2 Software Architecture Development Capability Detection

It is important to detect situations where a project lacks SA development capability, as these scenarios may inform actions to improve the project outcomes. To this end, we examine the SA and SE literature to deduce a set of socio-technical factors that impact SA quality. We map these to the PHIS literature to determine the final set of factors. We adopt an iterative approach based on an initial analysis of the literature. This approach consists of the following steps:

A. Identify factors that characterise the socio-technical impacts on SA.

B. Validate and enhance the set of factors identified in step A by comparing the derived set of factors with those identified in PHIS literature.

C. Develop and validate a factor rating scheme.

    1. Develop a rating scheme to determine the satisfaction of conditions relating to the identified factors.

    2. Validate and enhance the factor rating scheme developed in step 1 as follows:

        i. Determine the adequacy of the factor set definitions and associated rating scale by applying the factors to an exemplar of PHIS project failure.

        ii. Determine the reliability of the rating scales by applying the factors to further exemplars of PHIS project failure.

3.3 Factor Rating Scheme

We develop a rating scheme consisting of specific satisfaction levels for each factor and an associated rating scale. A rating scheme should be comprehensible to the Framework Users (e.g. Software Architects, Technical Project Managers and technical individuals within government organisations). The Framework User must identify a factor's presence (or degree) in the PHIS project. For example, to determine the pervasiveness of domain knowledge within the SE team, we may state that *domain knowledge exists sufficiently within the SE team*. A resulting rating scheme would manifest as:

Strongly Disagree – The domain knowledge **is not** sufficient
Disagree – The domain knowledge **is likely not** sufficient
Neutral – The domain knowledge sufficiency is **unknown**
Agree – The domain knowledge **is likely** sufficient
Strongly Agree - The domain knowledge **is** sufficient

In addition, the assessment should highlight factor deficiencies to enable further investigation. For example, when the SE team lacks expertise in implementing and testing SA. The validation of such a rating scheme would include its understandability and ease of use.

3.4 Capability Assessment Score and Grade

A final capability assessment score should be provided once the conditions pertaining to the factors have been rated. A comprehensible grading should be provided to accommodate the foreseeable possibility that Framework Users may present the grading to other non-technical individuals (e.g. government officials and politicians).

If all the factors are rated as Strongly Agree (factor is sufficiently satisfied), it suggests that the project capability is High. Similarly, if all the factors are rated as Agree (factor is likely to be sufficiently satisfied), it suggests that the project capability may be Moderate. Per contra, if all the factors are rated Strong Disagree (factor is not sufficiently satisfied) or Disagree (factor is likely not sufficiently satisfied), it suggests that the project capability is Very Low or Low, respectively. A somewhat uniform rating distribution or centrally clustered distribution may indicate that the capability is unknown/unclear. This grade is informative, suggesting that the factors should be addressed before continuing with the project.

3.5 Exemplars of Public Health Information Systems Failure

To evaluate the framework, it is imperative to assess its efficacy in scenarios where its performance is anticipated. To this end, exemplars of PHIS failure were chosen, where the failure prompted the production of a government audit report (an unstructured data source). Such reports provide valuable insight and perspectives into project failures, including observations of the organisation, project management, communication, and vendor performance.

The following criteria are selected to determine whether a chosen system was sufficient as an exemplar of PHIS failure:

1. The system has been launched (made available for real-world use)
2. The system has substantially poor end-user outcomes pertaining to issues such as instability, unreliability, lack of availability/access, poor performance, and data inconsistency.
3. The system's failures have prompted the production of a government audit report, which indicates severe problems with the system, such as instability, unreliability, lack of availability/access, poor performance, and data inconsistency.

4 Framework Synthesis

In the previous section, we outlined a set of outcomes that characterise the capability assessment framework. In this section, we delineate the iterative DSR build. At each iteration of the DSR build cycle, the framework was validated and evaluated, ensuring it would function correctly within the conceptual framework and design principles for which it was designed (see Section 3).

4.1 Iteration 1 - Synthesis of Preliminary Factors

Factors that characterise the socio-technical impacts on SA were identified. The synthesis utilised literature that proposed mitigations to Technical Debt and Architectural Technical Debt. For example, the grounded theory of Architectural Technical Debt (Verdecchia et al. 2021). The mitigations were categorised according to our model (see Appendix 1). We synthesised these mitigations into concise factor items in the form of conditions to be satisfied. Technical Debt papers that proposed mitigations (Freire 2020; Ramač et al. 2022; Rios et al. 2018) were included. Indeed, similar causes and mitigations appear in Architectural Technical Debt and Technical Debt papers, mapping coherently to our conceptual model. We then examined Architectural Technical Debt and Technical Debt papers that named causal factors that mapped to our model. For example, the cause "lack of code reviews" mapped to the Practice category of our model. We deduced that the antithesis of "lacking code reviews" was to *conduct sufficient effective code reviews* – A mitigation. Hence, we added these to our factor items. Finally, the SA literature was also examined to extract additional factors. The final set of factor items was assigned a preliminary desired state:

Exists - The item is present.

Identified - The item has been identified.

Defined - The item has been explicitly defined.

Understood - The item's comprehensibility has been explicitly determined.

Two exemplars of PHIS failure were selected in the initial validation:

Exemplar 1 - A payroll system (Queensland, Australia) (Chesterman 2013). When the system was launched, it had severe financial consequences for hospital staff. They were underpaid, overpaid, and even not paid at all. It failed to handle contract permutations; hence, data corruption was apparent.

Exemplar 2 - A clinical Information and Communications Technology hospital system (Victoria, Australia) (Doyle 2013). Among many problems, the system incorrectly considered patients as having been discharged when they were transferred to other wards. Patients were also potentially at risk of incorrect/missing medication doses.

The validation of the theoretical constructs was conducted utilising Exemplar 1. The report was reviewed to extract pertinent information pertaining to the factor items' constructs (see categories in Appendix 1). The extracted data was labelled with the appropriate factor category and then synthesised. Inferences pertaining to the circumstances and events within the projects were documented. Each factor item was marked with one of the predefined states according to the researcher's understanding of the reports. The evaluation procedure was comprised of the following:

For each factor in the framework,

1. Record excerpts that delineate or imply the conditions applicable to the factor.
2. Record one of the predefined states.

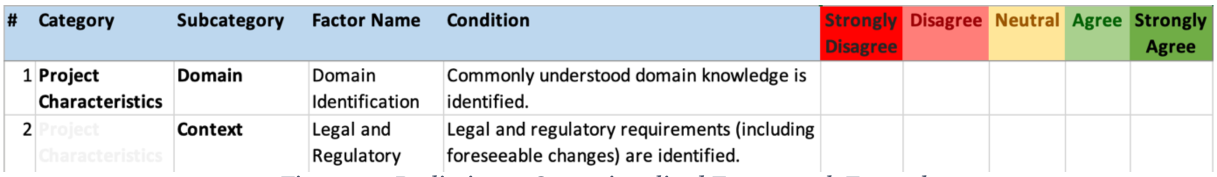
Where there is no apparent text regarding the factor item in question, inferences are drawn based on the available information. If no reasonable inference can be made, a neutral/unknown label is assigned.

The framework detected failure when given the conditions in the report. This initial validation of the theoretical constructs required confirmation. To this end, the framework was then applied to Exemplar 2, following the same procedure. Again, the framework detected failure.

The output of this iteration was a preliminary framework containing a list of factors (*n=74*) contributing to SA failure/success. This preliminary framework provided evidence that the framework was efficacious in detecting poor SA outcomes.

4.2 Iteration 2 – Operationalisation of the framework

The factor items were reformulated into conditions and presented as statements about the conditions. The factor items were also refined to facilitate the assignment of ratings. That is the factors identified as potentially ambiguous (or with low comprehensibility) were subsequently rephrased. The rating scale was created as described in Section 3.3. The factors were organised and presented to show the category, subcategory, factor name, and condition presented as a statement for evaluation. An excerpt of the factors and the rating scale are shown in Figure 1. The figure shows two factors.



#	Category	Subcategory	Factor Name	Condition	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
1	Project Characteristics	Domain	Domain Identification	Commonly understood domain knowledge is identified.					
2	Project Characteristics	Context	Legal and Regulatory	Legal and regulatory requirements (including foreseeable changes) are identified.					

Figure 1 – Preliminary Operationalised Framework Example

The usefulness of the ratings was enhanced with a numerical component to facilitate the calculation of an overall capability score (see Table 1). A simple sum is calculated using the weightings to provide a total weighted score. The procedure for calculating the capability score is as follows:

1. *Column count* - For each rating column, sum the number of responses.
2. *Weighted column score* - Multiply the column count by the weighting for that column.
3. *Total weighted score* - Sum the weighted column counts.

Rating	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
Score	-2	-1	0	1	2

Table 1 - Rating Scale Weightings

The validation and evaluation were conducted utilising the exemplars from Iteration 1 (see Section 4.1). This ensured the framework's repeatability and efficacy while providing a test for the rating scale. Each factor item statement was assessed for appropriateness, comprehensibility and answerability for the

Framework Users. To this end, personas were adopted to perform the evaluation. Namely, that of a Technical Project Manager and a Software Architect. *Note that the primary researcher is an industry expert with experience in SE, SA, technical leadership, and project management.* Thus, the features of the adopted personas are grounded in practice. Specifically, they are assumed to have knowledge of the end-to-end software development lifecycle, including quality assurance practices within the SE team. In the case of a government organisation, we assume that the government organisation will have individuals with a sufficient understanding of the technical aspects of the project. Such individuals may reside within the larger organisation or perhaps in an adjacent or overarching IT organisation.

The following questions were asked about the statements:

1. Does the persona have sufficient knowledge and experience to understand the statement?

2. Can a rating response be selected for the statement?

3. Is there ambiguity in the statement which may confuse the responder?

The evaluation procedure was comprised of the following:

For each factor in the framework,

1. Record excerpts that delineate or imply the conditions applicable to the rating.
2. Record a rating.
3. Record a rationale for the above rating.

Where there is no explicit text pertaining to the factor item, then inferences may be made. Where there is no feasible inference, then a neutral rating is given.

The completed factor ratings and capability assessment scores represent the validation and evaluation of the theoretical constructs and the framework. The statements facilitated ratings appropriately. The capability score provided an indication of failure in the scenarios. The comprehensibility of the statements was open for debate among the authors. For example, a Technical Project Manager may require further elaboration/instructions to understand the statements, whereas a Software Architect may not. This potential need for supplementary information will be addressed in the next section.

The output of this iteration was a set of reformulated factor items and a rating scale for the factors contributing to SA's success/failure.

4.3 Iteration 3 – Final Iteration

The comprehensibility of the factors was addressed by providing supplementary information in the form of assertions. The refinement of the factors illuminated the necessity to reorganise and consolidate some factors between categories. For example, issues pertaining to stakeholders were moved from the Project Characteristics category to the Organisation category. Similarly, some sub-categories were integrated and reorganised to facilitate further comprehensibility. Importantly, some factors were also consolidated to improve comprehensibility and conciseness.

A *weighted* score range of -136 to 136 is calculated using the number of consolidated factors (*n=68*) and the weightings in Table 1. To increase the comprehensibility of the capability assessment, the weighted score is mapped into a corresponding score in the range of 0 to 100. A capability grade is provided by mapping the number to a grade: High, Moderate, Unknown/Unclear, Low, and Very Low. These are shown in Table 2.

Capability Grade	Lower Weighted Points	Upper Weighted Points	Lower Capability Score (out of 100)	Upper Capability Score (out of 100)
High	82	136	80.15	100.00
Moderate	28	81	60.29	79.78
Unknown/Unclear	-28	27	39.71	59.93
Low	-82	-29	19.85	39.34
Very Low	-136	-83	0.00	19.49

Table 2 - Capability Assessment Ranges

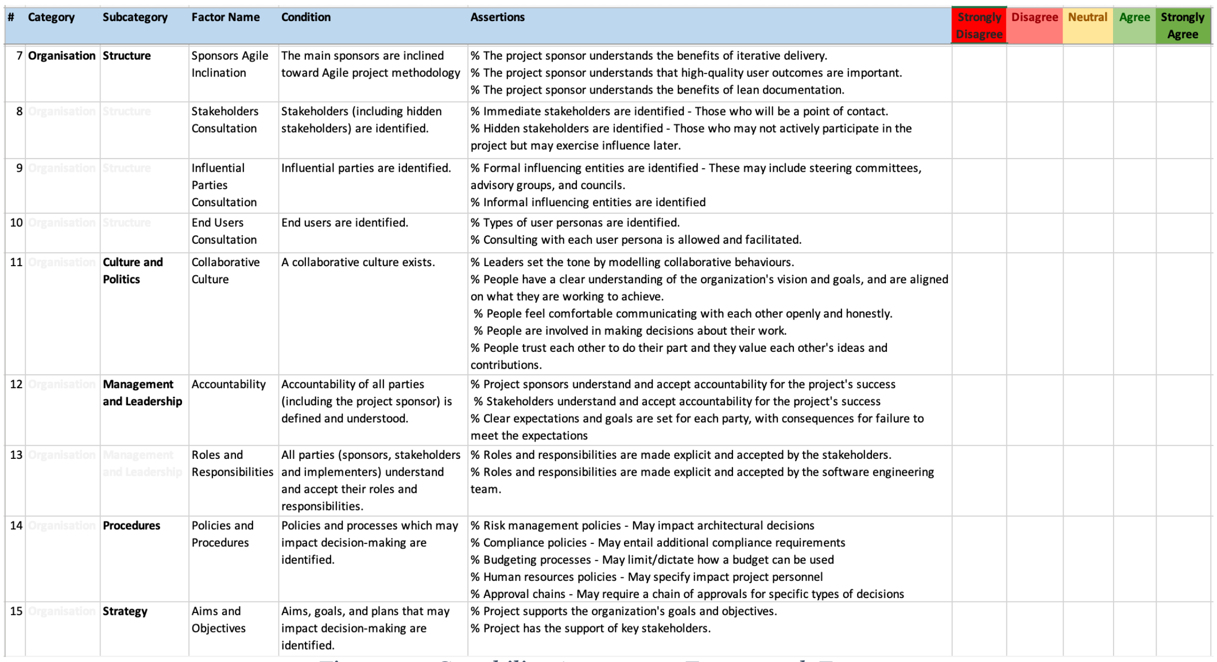
The validation and evaluation were conducted utilising the exemplars from Iteration 1 (see Section 4.1). Again, this ensured the framework's repeatability and efficacy while providing a test for the rating scale and grade. As per the previous iteration, the personas of a Technical Project Manager and a Software Architect were adopted. The capability score continued to indicate failure in the scenarios. The comprehensibility changes facilitated the assignment of ratings; however, further rephrasing of the factors followed this evaluation.

Finally, an industry expert (with SA experience) was consulted to validate the framework's comprehensibility and usefulness at face value. The expert was provided with a random set of factor items to validate yes/no for efficacy. The industry expert confirmed that the items were efficacious. The expert also confirmed that a random set of inferences and assumptions made about the report were valid and logically deducted. The expert confirmed that the framework appeared easy to use.

The output of this iteration is the elaboration of factor conditions, a consolidation of the factors (*n=68*), and capability score and grade. The satisfaction of design principles (outlined in Section 3.1) thus concluded the iterative DSR process. The output is hence considered Version 1 of the framework.

4.4 The Capability Assessment Framework

An excerpt of the final framework is shown in Figure 2. The figure shows the Organisation category of the framework. In this category, there are five subcategories and nine factors. The design principles (described in Section 3.1) underpinned the framework's design. They are described as follows.



#	Category	Subcategory	Factor Name	Condition	Assertions	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
7	**Organisation**	**Structure**	Sponsors Agile Inclination	The main sponsors are inclined toward Agile project methodology	% The project sponsor understands the benefits of iterative delivery. % The project sponsor understands that high-quality user outcomes are important. % The project sponsor understands the benefits of lean documentation.					
8	Organisation	Structure	Stakeholders Consultation	Stakeholders (including hidden stakeholders) are identified.	% Immediate stakeholders are identified - Those who will be a point of contact. % Hidden stakeholders are identified - Those who may not actively participate in the project but may exercise influence later.					
9	Organisation	Structure	Influential Parties Consultation	Influential parties are identified.	% Formal influencing entities are identified - These may include steering committees, advisory groups, and councils. % Informal influencing entities are identified					
10	Organisation	Structure	End Users Consultation	End users are identified.	% Types of user personas are identified. % Consulting with each user persona is allowed and facilitated.					
11	Organisation	**Culture and Politics**	Collaborative Culture	A collaborative culture exists.	% Leaders set the tone by modelling collaborative behaviours. % People have a clear understanding of the organization's vision and goals, and are aligned on what they are working to achieve. % People feel comfortable communicating with each other openly and honestly. % People are involved in making decisions about their work. % People trust each other to do their part and they value each other's ideas and contributions.					
12	Organisation	**Management and Leadership**	Accountability	Accountability of all parties (including the project sponsor) is defined and understood.	% Project sponsors understand and accept accountability for the project's success % Stakeholders understand and accept accountability for the project's success % Clear expectations and goals are set for each party, with consequences for failure to meet the expectations					
13	Organisation	Management and Leadership	Roles and Responsibilities	All parties (sponsors, stakeholders and implementers) understand and accept their roles and responsibilities.	% Roles and responsibilities are made explicit and accepted by the stakeholders. % Roles and responsibilities are made explicit and accepted by the software engineering team.					
14	Organisation	**Procedures**	Policies and Procedures	Policies and processes which may impact decision-making are identified.	% Risk management policies - May impact architectural decisions % Compliance policies - May entail additional compliance requirements % Budgeting processes - May limit/dictate how a budget can be used % Human resources policies - May specify impact project personnel % Approval chains - May require a chain of approvals for specific types of decisions					
15	Organisation	**Strategy**	Aims and Objectives	Aims, goals, and plans that may impact decision-making are identified.	% Project supports the organization's goals and objectives. % Project has the support of key stakeholders.					

Figure 2 – Capability Assessment Framework Excerpt

Comprehensiveness - The identified factors are socio-technical in nature, covering a large breadth of inputs and processes of a project. The factors are grouped by category and sub-category. A condition is named for each factor, in addition to assertions (observations/circumstances) that may indicate the satisfaction of the condition. The categories may be seen in Appendix 1.

Simplicity and Clarity – The factors are organised by category and subcategory to provide context to the Framework User. The factors are presented as conditions to be sufficiently satisfied. The statements pertaining to the conditions are designed to be understandable by Technical Project Managers, Software Architects, and technical people within government organisations. In addition, assertions are provided for each factor to help guide the Framework User. A score (0 to 100) and human-comprehensible grade are provided.

Reliability – The framework is derived from SE literature, containing Architectural Technical Debt and Technical Debt causes and mitigations, and then mapped to PHIS literature. The synthesis was conducted utilising exemplars of PHIS failure, indicating that its ability to detect SA development capability is applicable across PHIS projects.

User-Centred Design - The artifact provides a visual colour scheme, indicating the capability of each factor to the Framework User. Simply put, light green and dark green are positive, light red and dark red are negative, and orange indicates ambiguity (see Figure 2). The applied framework is shown in Figure 3.

Actionability – The capability grade allows the Framework User to determine if a project should be continued with the current set of circumstances. When a factor is not sufficiently satisfied, the Framework User may use this as a data point for investigation and improvement of the circumstances pertaining to that factor. The provided factor assertions may facilitate these actions for improvement.

In the following section, we evaluate and discuss the framework.

5 Evaluation and Discussion

In this section, we evaluate the final framework, followed by a discussion of the results and implications.

5.1 Evaluation

This evaluation determines the efficacy of the framework to detect failure scenarios. That is, to ensure that the theoretical constructs in the framework are efficacious. We utilise an additional exemplar of SA failure in PHIS – healthcare.gov (Office 2014) – A health insurance marketplace. When healthcare.gov was launched, several critical and severe issues impacted its functionality. These issues included:

Instability - The website frequently crashed, with users being greeted by error messages and non-functioning pages.

Poor scalability and *unavailability* - The high volume of users attempting to access the system resulted in long downtimes.

Data Inconsistency - Data transmitted between other systems was often corrupted or incomplete. This led to incorrect eligibility determinations and incorrect subsidy calculations. In addition, user data was incorrectly saved, where some data was only partially saved, resulting in lost data.

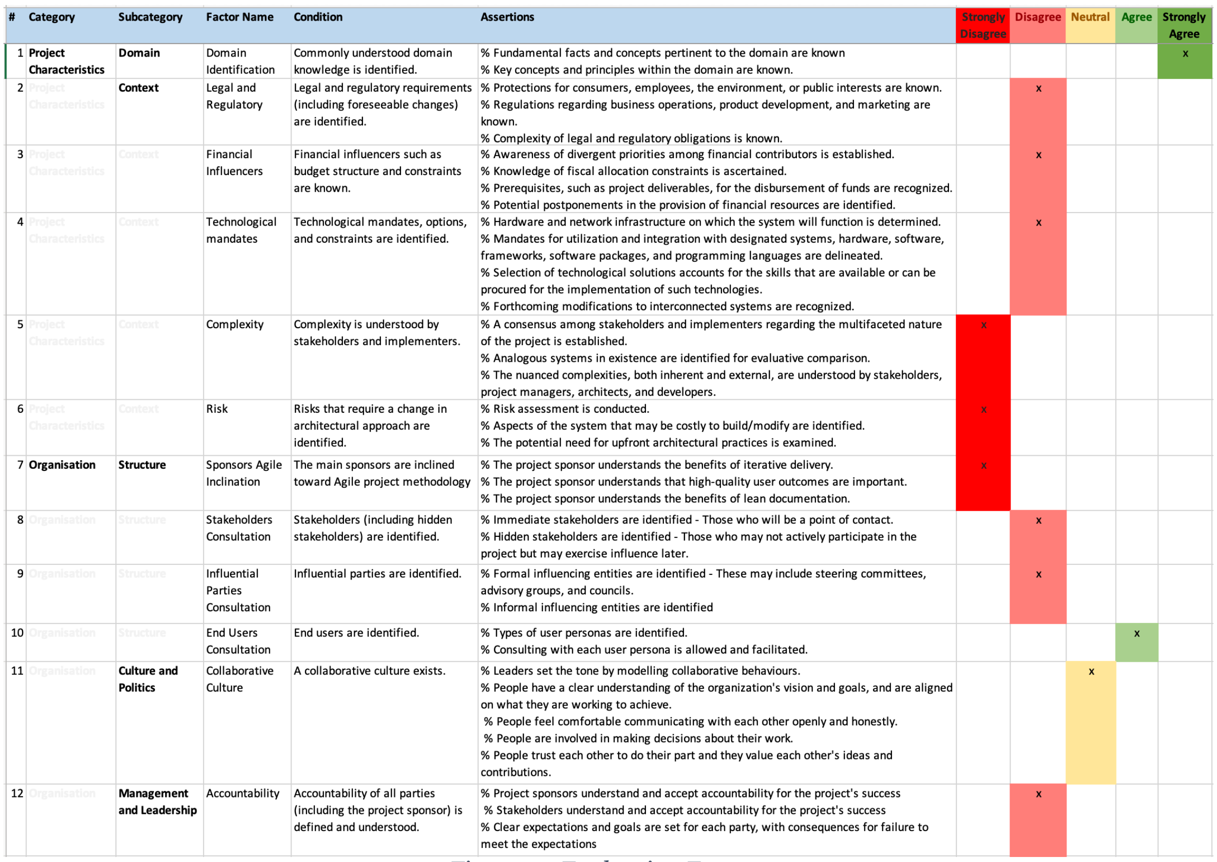
Various circumstances lead to these problems, including insufficient planning, ineffective oversight practice, insufficient identification of risks, commencing the development before regulatory requirements were finalised or the number of participating states was known, and a lack of established experience in Agile project management methodology. Applying the framework to this exemplar follows the procedure established in Section 4.1. The evaluation confirms that the framework continues to satisfy the requirement to detect a failure scenario. An excerpt of the evaluation is shown in Figure 3. The excerpt shows apparent deficiencies in the project pertaining to the Project Characteristics and Organisation. The applied ratings addressed circumstances such as *the project commencing before regulations were in place, the project commencing before the number of participating states was known, and insufficient planning or risk considerations*. These circumstances potentially impact financial and technological concerns. Agile methodology was new to the government organisation, and many change requests indicate that the stakeholders were not sufficiently involved in defining the requirements. In Figure 4, the capability score and grade show that the framework could detect that the project's circumstances led to SA failure.

5.2 Discussion

In the previous sections, we delineated the iterative process for developing our framework and evaluated it utilising three exemplars of PHIS failure. The DSR procedure was advantageous, as it provided a structured approach to facilitate the systematic design and evaluation of the framework. The initial framework synthesis utilised literature across the multi-disciplinary fields of SE and PHIS. This focused the framework on the PHIS domain regarding known failure and success factors. The unstructured data source (government audit reports) provided a means of evaluating the framework's theoretical constructs against real-world cases. The framework performed well in all evaluations, detecting scenarios resulting in poor end-user outcomes. It appears that many factors are interrelated/interdependent. This confirms the importance of the framework in illuminating multi-dimensional factors in PHIS projects. Interestingly, the most prevalent factors in each PHIS project failure were pertaining to the Organisation, Requirements, and Project management.

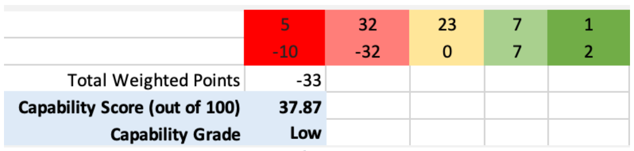
Our evaluation procedure utilised exemplars from PHIS failures. These involved differing government organisations, geographically distant, and differing system purposes and requirements. This provides evidence to support a generalisation of the problem from specific instances to the wider PHIS domain.

When government organisations utilise the framework, it may be applied during the tender process and at important project stages. Indeed, the framework may be utilised as a contractual agreement whereby vendors must continue to satisfy the framework's assessment grade. Vendor organisations may, in turn, use the framework to self-assess and to also assess their technology partners. It should be noted that government organisations will not necessarily have direct knowledge of the internal practices of their vendors. Therefore, they will likely rely on vendor self-reporting. Importantly, PHIS projects tend to be multi-year endeavours. During this time, many changes may occur to the organisations involved. Key individuals may leave the government organisation and/or vendors, leaving skills and knowledge gaps. New policies may also be introduced into law. A particular concern is when government departments are created, merged, or disbanded. Therefore, we anticipate that our framework could be applied periodically and when changes occur within the organisations.



#	Category	Subcategory	Factor Name	Condition	Assertions	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
1	**Project Characteristics**	**Domain**	Domain Identification	Commonly understood domain knowledge is identified.	% Fundamental facts and concepts pertinent to the domain are known % Key concepts and principles within the domain are known.					x
2	Project Characteristics	**Context**	Legal and Regulatory	Legal and regulatory requirements (including foreseeable changes) are identified.	% Protections for consumers, employees, the environment, or public interests are known. % Regulations regarding business operations, product development, and marketing are known. % Complexity of legal and regulatory obligations is known.		x			
3	Project Characteristics	Context	Financial Influencers	Financial influencers such as budget structure and constraints are known.	% Awareness of divergent priorities among financial contributors is established. % Knowledge of fiscal allocation constraints is ascertained. % Prerequisites, such as project deliverables, for the disbursement of funds are recognized. % Potential postponements in the provision of financial resources are identified.		x			
4	Project Characteristics	Context	Technological mandates	Technological mandates, options, and constraints are identified.	% Hardware and network infrastructure on which the system will function is determined. % Mandates for utilization and integration with designated systems, hardware, software, frameworks, software packages, and programming languages are delineated. % Selection of technological solutions accounts for the skills that are available or can be procured for the implementation of such technologies. % Forthcoming modifications to interconnected systems are recognized.		x			
5	Project Characteristics	Context	Complexity	Complexity is understood by stakeholders and implementers.	% A consensus among stakeholders and implementers regarding the multifaceted nature of the project is established. % Analogous systems in existence are identified for evaluative comparison. % The nuanced complexities, both inherent and external, are understood by stakeholders, project managers, architects, and developers.	x				
6	Project Characteristics	Context	Risk	Risks that require a change in architectural approach are identified.	% Risk assessment is conducted. % Aspects of the system that may be costly to build/modify are identified. % The potential need for upfront architectural practices is examined.	x				
7	**Organisation**	**Structure**	Sponsors Agile Inclination	The main sponsors are inclined toward Agile project methodology	% The project sponsor understands the benefits of iterative delivery. % The project sponsor understands that high-quality user outcomes are important. % The project sponsor understands the benefits of lean documentation.	x				
8	Organisation	Structure	Stakeholders Consultation	Stakeholders (including hidden stakeholders) are identified.	% Immediate stakeholders are identified - Those who will be a point of contact. % Hidden stakeholders are identified - Those who may not actively participate in the project but may exercise influence later.		x			
9	Organisation	Structure	Influential Parties Consultation	Influential parties are identified.	% Formal influencing entities are identified - These may include steering committees, advisory groups, and councils. % Informal influencing entities are identified		x			
10	Organisation	Structure	End Users Consultation	End users are identified.	% Types of user personas are identified. % Consulting with each user persona is allowed and facilitated.				x	
11	Organisation	**Culture and Politics**	Collaborative Culture	A collaborative culture exists.	% Leaders set the tone by modelling collaborative behaviours. % People have a clear understanding of the organization's vision and goals, and are aligned on what they are working to achieve. % People feel comfortable communicating with each other openly and honestly. % People are involved in making decisions about their work. % People trust each other to do their part and they value each other's ideas and contributions.			x		
12	Organisation	**Management and Leadership**	Accountability	Accountability of all parties (including the project sponsor) is defined and understood.	% Project sponsors understand and accept accountability for the project's success % Stakeholders understand and accept accountability for the project's success % Clear expectations and goals are set for each party, with consequences for failure to meet the expectations		x			

Figure 3 - Evaluation Excerpt



	5	32	23	7	1
	-10	-32	0	7	2
Total Weighted Points	-33				
Capability Score (out of 100)	**37.87**				
Capability Grade	**Low**				

Figure 4 - Evaluation Summary

Some degree of interpretation may be required for capability scores near the lower or upper limits of the provided ranges. For example, as shown in Table 2, a score of 59/100 corresponds to an Unknown/Unclear capability, whereas a score of 60/100 corresponds to a Moderate capability. Given that the framework is presented as a colourised questionnaire, *when the Framework User completes their PHIS project assessment, the framework is effectively presented as an actionable factor-level capability report* (as can be seen in Figure 3). Therefore, it is recommended that the framework's capability report always be examined for potential factor-level deficiencies.

5.3 Limitations

PHIS failure reports tend to focus on project governance, management, and organisational factors, not detailed software development practices. Therefore, some inferences and assumptions were necessary to extrapolate events and circumstances within the projects. These inferences may be subject to human bias. We mitigated this by utilising an industry expert to verify a random set of these assumptions. When manually processing the reports, there is a potential for human error and bias. This was mitigated by performing two iterations of the manual processing. The synthesis was also checked twice, and the results were discussed with the secondary researcher. To mitigate human error and bias, future work could involve using AI for evaluation.

While the evaluation confirmed the theoretical validity of the factor constructs, the framework was evaluated post hoc – i.e., an already completed PHIS project. This may pose questions regarding the efficacious value of the framework in a pre-PHIS project or as a monitoring tool during the project. *However, the framework conditions are manifested in the present tense. It follows that the framework evaluates conditions at the time of applying the framework.* These conditions were derived from mitigation recommendations in the literature. Further evaluation may still be required for success scenarios.

While the researchers thoroughly discussed the design principles, further study may be required to assess how well the framework satisfied the principles. The perceived usefulness of the framework may be influenced by human subjectivity. For example, the sufficient use of test cases may differ between individuals. The vendor may deem their practice as sufficient, whereas the government organisation may view the vendor's practice as insufficient. We do not prescribe a specific discrete level of what is meant as “sufficient” or “effective”. We would recommend defaulting to industry best practices and proven practices to support these assertions.

6 Conclusion and Future Work

Addressing SA development success factors within PHIS projects is necessary to improve end-user outcomes. *In this paper, we presented Capable - a framework for assessing the SA development capability of PHIS projects.* Our novel contribution utilised DSR to synthesise the framework from the existing literature. The framework was empirically evaluated using exemplars of Australian and American PHIS project failures. The framework successfully detected failure scenarios while providing an indicative capability grade and score, as well as an actionable factor-level capability report. This work is an important first step to help ensure that PHIS projects are capable of developing appropriate software architectures - A real-world problem. Indeed, the framework is intended to be applied before software development commences and continually assess capability throughout the project lifecycle. The Framework Users may represent individuals from both government organisations and their vendors. Thus, the contribution of this research (to both the literature and practice) can be summarised as follows:

Individuals such as Chief Technical Officers, Technical Project Managers, Software Architects, and technical individuals within government organisations may use the framework to assess the capability of developing Software Architecture in Public Health Information Systems projects before and during the project lifecycle.

Future work will further examine the use of the framework in the field and further evaluate its efficacy, specifically in success scenarios. Human subjectivity, when applying the framework, may necessitate further examination. Guidance may be required when a vendor and government organisation cannot agree on a satisfaction rating. Given that the framework’s factors were derived from Technical Debt, Architectural Technical Debt mitigations, and IS success factors, the framework may be applicable more generally. There may be some scope for trimming the framework to some core factors to enable the development of a lightweight version. It may be speculated that some individual factors may disproportionately impact the resulting SA. This warrants further investigation. Given the framework’s ease of use and grounding in SE, it is also hoped that its applicability can be generalised to other domains. Future evaluations (including generalisability) may also include the use of AI.

7 References

Bass, L., Clements, P., and Kazman, R. 2021. *Software Architecture in Practice*, Addison-Wesley Professional.

Chesterman, R. N. 2013. "Queensland Health Payroll System Commission of Inquiry," *Queensland Health Payroll System Commission of Inquiry*, Queensland Health Payroll System Commission of Inquiry, July. (http://www.healthpayrollinquiry.qld.gov.au/__data/assets/pdf_file/0014/207203/Queensland-Health-Payroll-System-Commission-of-Inquiry-Report-31-July-2013.pdf).

Dang, A., and Beydoun, G. 2023. "Toward Addressing the Software Architecture Blind Spot of Information System Success Factors in the Public Health Domain," in *Australasian Conference on Information Systems*.

Demir, M. Ö., Chouseinoglou, O., and Tarhan, A. K. 2024. "Factors Affecting Architectural Decision-Making Process and Challenges in Software Projects: An Industrial Survey," *Journal of Software: Evolution and Process*, Wiley Online Library, p. e2703.

Douglas, G. 2021. "An Insider's Perspective: Governance of Large ICT Software Projects in the Australian and New Zealand Public Sectors," PhD Thesis, PhD Thesis, The Australian National University (Australia).

Doyle, J. 2013. "Victorian Auditor-General's Report Clinical ICT Systems in the Victorian Public Health Sector," *Victorian Auditor-General's Report*, Victorian Auditor-General's Office. (https://www.audit.vic.gov.au/sites/default/files/20131030-Clinical-ICT-Systems.pdf).

Eden, R., and Sedera, D. 2014. "The Largest Admitted IT Project Failure in the Southern Hemisphere: A Teaching Case," in *Proceedings of the 35th International Conference on Information Systems*, Association for Information Systems (AIS), pp. 1–15.

Freire, R. O., Salvio, Rios, Nicolli, Mendonça, Manoel, Falessi, Davide. ,. Seaman, Carolyn, Izurieta, Clemente. ,. Spínola. 2020. "Actions and Impediments for Technical Debt Prevention: Results from a Global Family of Industrial Surveys," *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, pp. 1548–1555.

Galster, M., Tamburri, D. A., and Kazman, R. 2017. "Towards Understanding the Social and Organizational Dimensions of Software Architecting," *ACM SIGSOFT Software Engineering Notes* (42:3), ACM New York, NY, USA, pp. 24–25.

Gregor, S., and Hevner, A. R. 2013. "Positioning and Presenting Design Science Research for Maximum Impact," *MIS Quarterly*, JSTOR, pp. 337–355.

Institute, C. 2024. *CMMI V3.0: Capability Maturity Model Integration*.

ISO. 2015. *ISO 9001:2015 - Quality Management Systems -- Requirements*. (https://www.iso.org/standard/62085.html).

Kaplan, B., and Harris-Salamone, K. D. 2009. "Health IT Success and Failure: Recommendations from Literature and an AMIA Workshop," *Journal of the American Medical Informatics Association* (16:3), BMJ Group BMA House, Tavistock Square, London, WC1H 9JR, pp. 291–299.

Li, E. Y. 1997. "Perceived Importance of Information System Success Factors: A Meta Analysis of Group Differences," *Information & Management* (32:1), Elsevier, pp. 15–28.

Ludlow, M. 2016. "IT Disasters Now Part of Modern Life," *Australian Financial Review*. (https://www.afr.com/technology/it-disasters-now-part-of-modern-life-20160628-gptyw6).

McManus, J., and Wood-Harper, T. 2007. "Understanding the Sources of Information Systems Project Failure," *Journal of the Management Services Institute*.

Nielsen, J. 1994. *Usability Engineering*, Morgan Kaufmann.

Norman, D. A., and Draper, S. W. 1986. *User Centered System Design; New Perspectives on Human-Computer Interaction*, L. Erlbaum Associates Inc.

Office, U. S. G. A. 2014. "Ineffective Planning and Oversight Practices Underscore the Need for Improved Contract Management," *Report to Congressional Requesters*, , July. (https://www.gao.gov/assets/gao-14-694.pdf).

Ombudsman, N. S. W. 2024. "Machinery of Government Changes and Maladministration Risks," *In Focus*. (https://www.ombo.nsw.gov.au/__data/assets/pdf_file/0010/145000/Machinery-of-government-changes-and-maladministration-risks.pdf).

Petter, S., DeLone, W., and McLean, E. R. 2013. "Information Systems Success: The Quest for the Independent Variables," *Journal of Management Information Systems* (29:4), Taylor & Francis, pp. 7–62.

Ramač, R., Mandić, V., Taušan, N., Rios, N., Freire, S., Pérez, B., Castellanos, C., Correal, D., Pacheco, A., Lopez, G., and others. 2022. "Prevalence, Common Causes and Effects of Technical Debt: Results from a Family of Surveys with the IT Industry," *Journal of Systems and Software* (184), Elsevier, p. 111114.

Rios, N., Spinola, R. O., de Mendonça Neto, M. G., and Seaman, C. 2018. "A Study of Factors That Lead Development Teams to Incur Technical Debt in Software Projects," *2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, IEEE, pp. 429–436.

Siau, K., Long, Y., and Ling, M. 2010. "Toward a Unified Model of Information Systems Development Success," *Journal of Database Management (JDM)*, pp. 80–101. (https://doi.org/10.4018/jdm.2010112304).

Silvius, A. G., and Schipper, R. 2015. "A Conceptual Model for Exploring the Relationship between Sustainability and Project Success," *Procedia Computer Science* (64), Elsevier, pp. 334–342.

Soliman, M., Avgeriou, P., and Li, Y. 2021. "Architectural Design Decisions That Incur Technical Debt—An Industrial Case Study," *Information and Software Technology* (139), Elsevier, p. 106669.

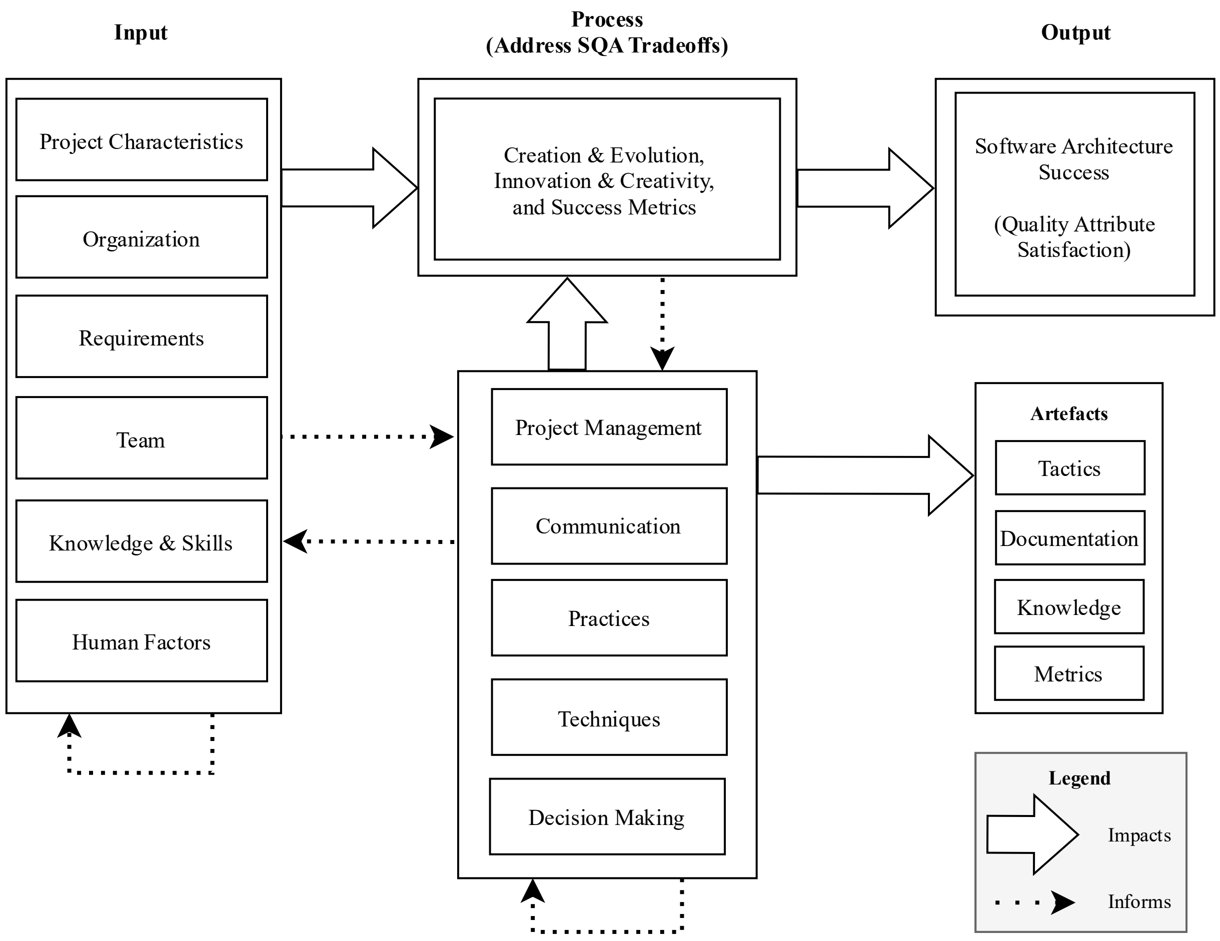
Vaishnavi, V., and Kuechler, W. 2004. *Design Research in Information Systems*. (http://www.desrist.org/design-research-in-information-systems/).

Varajão, J., Lourenço, J. C., and Gomes, J. 2022. "Models and Methods for Information Systems Project Success Evaluation–A Review and Directions for Research," *Heliyon* (8:12), Elsevier.

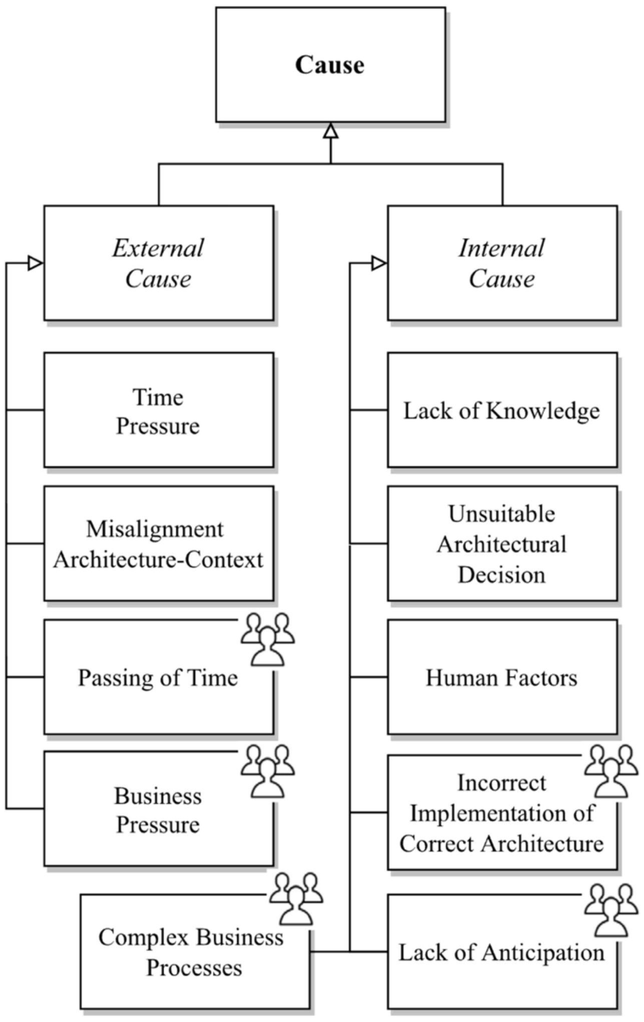
Verdecchia, R., Kruchten, P., Lago, P., and Malavolta, I. 2021. "Building and Evaluating a Theory of Architectural Technical Debt in Software-Intensive Systems," *Journal of Systems and Software* (176), Elsevier, p. 110925.

Yeo, K. T. 2002. "Critical Failure Factors in Information System Projects," *International Journal of Project Management* (20:3), Elsevier, pp. 241–246.

Appendix 1 - The Capability Assessment Framework High-level Categories from (Dang and Beydoun 2023)



Appendix 2 - Overview of Architectural Technical Debt Causes (Verdecchia et al. 2021)


Cause
External Cause
Internal Cause
Time Pressure
Misalignment Architecture-Context
Passing of Time
Business Pressure
Complex Business Processes
Lack of Knowledge
Unsuitable Architectural Decision
Human Factors
Incorrect Implementation of Correct Architecture
Lack of Anticipation

Acknowledgements

This research is supported by an Australian Government Research Training Program Scholarship.