A Framework for Considering Uncertainty in Software Systems

Chawanangwa Lupafya
School of Computer Science
University of St Andrews
St Andrews, United Kingdom
0000-0002-9698-1477

Dharini Balasubramaniam
School of Computer Science
University of St Andrews
St Andrews, United Kingdom
0000-0002-5093-0906

***Abstract*—There are many aspects involved in the development and operation of a software system, including system artefacts, activities, and infrastructure. Most of these aspects are vulnerable to uncertainty, which can result in risks to system quality and performance. Thus it is important to identify, represent and manage uncertainty in software systems. We hypothesise that using an underlying conceptual framework for characterising uncertainty can facilitate these activities. This paper demonstrates the use of an extensible framework, which defines a foundation for the systematic and explicit consideration of uncertainty in software systems. A software architecture case study is used to illustrate and evaluate the framework. A discussion of potential uses for the framework and future research is also provided.**

***Index Terms*—Uncertainty, Framework, Uncertainty in Software, Software Architecture**

I. Introduction

Uncertainty is an inherent feature of software systems. While uncertainty can offer opportunities, it can also have negative consequences. It is therefore important to consider uncertainty as a first-class concern during the development and operation of systems and attempt to anticipate and mitigate its impact [1].

This paper describes an effort to develop a generic conceptual framework for considering uncertainty in software systems. The framework can be used at different stages, such as requirements elicitation, design, implementation and quality assurance, and for different artefacts, such as requirements or user stories, design documents and evaluation scenarios. The resulting uncertainty data can be used for analysis, evaluation and decision making [2].

The novel contributions of this work include the definition of a conceptual framework for uncertainty in software systems, support for the application of this framework to software architectures via a workbench implemented by extending an existing open-source diagramming software, and the use of a case study from NASA's IMPALA architecture [1] to demonstrate and evaluate the framework.

The rest of the paper is structured as follows. Section II discusses the motivation for defining the uncertainty framework. Section III describes related work. Section IV presents our approach, which also outlines the process of defining the framework, and Section V summarises the framework. Section VI provides an overview of the workbench as a proof of concept implementation. Section VII presents the case study. Finally, a discussion of the case study and conclusions are presented in Sections VIII and IX.

II. Motivation

In day to day life, uncertainty is a common and familiar concept with many dimensions. For instance, the Oxford English Dictionary[2] (OED) defines uncertainty as "the quality of being uncertain in respect of duration, continuance, occurrence, etc.; liability to chance or accident. Also, the quality of being indeterminate as to magnitude or value; the amount of variation in a numerical result that is consistent with observation".

In informal circumstances, an intuitive grasp of the essence of the uncertainty is usually sufficient. However, in software systems, particularly those for which there are stringent quality requirements, a precise understanding of uncertainties and their consequences can be important. If uncertainty is not examined in detail, valuable data that can help uncertainty management may not be available.

For example, when we consider the reliability of a system component, we might use probability to represent the uncertainty relating to component failure. While this might be adequate for measurement purposes, the uncertainty has other attributes, which, if systematically recorded, can provide useful information to help manage it. In this scenario, in addition to

[1] https://ntrs.nasa.gov/citations/20160011412
[2] https://www.oed.com/

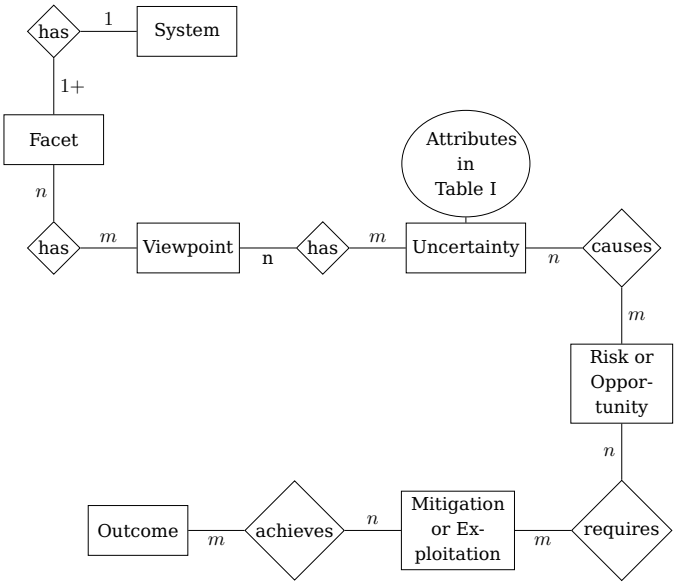

Fig. 1. Conceptual model: characterisation of uncertainty in systems

the measure of the uncertainty, stakeholders may be interested in details such as the underlying cause of the uncertainty, the lifespan of the uncertainty, any risks that arise from the uncertainty and possible mitigations for the uncertainty. In this paper, we present an extensible and customisable framework for characterising uncertainty with its multiple attributes.

III. Related Work

Uncertainty can be broadly classified as either epistemic or aleatory [3]. Epistemic uncertainty is due to lack of knowledge while aleatory uncertainty is due to inherent variability. There is existing work that conceptualises uncertainty in specific domains or contexts, such as in policy modelling [4], self-adaptive systems [5]–[8], complex systems [9], [10], cyber-physical systems [11], [12] and cognitive science [3]. Our approach builds on these existing frameworks.

IV. Our approach

The proposed framework consolidates, generalises and extends existing uncertainty concepts so that they can be customised and used regardless of context. A case study is used to evaluate the framework through demonstrating its use for identify uncertainties in the software architecture views of a system.

Prior to defining the framework, we conducted a literature search of existing frameworks for explicitly characterising uncertainty. The search was conducted in the following academic databases: IET Digital Library[3], Web of Science[4], DBLP[5], Scopus[6], ACM Library[7], and IEEE Xplore[8]. The search terms we used on the databases were: *'uncertainty AND (software OR system) AND framework'*. We then conducted a manual search to obtain more comprehensive results by following up references from results returned by the search. We analysed the search results manually to identify primary references, which either propose or consolidate concepts for characterising uncertainty.

The identified approaches consider uncertainty in the following contexts. Cognitive science is related to human thought and behaviour [3]. There is a strong intertwined relationship between physical and software components in cyber-physical systems [11]. Complex systems comprise numerous complex components, interconnections and dynamism [9]. Self-adaptive systems automatically adapt to counter the influence of uncertainties against performance goals [13], [14].

V. The Framework

The conceptual model resulting from the definition process is shown as an ER diagram (ERD) in Figure 1. The framework includes the software system, its facets, which are particular aspects in which uncertainty can be incorporated such as artefacts and processes, and viewpoints, which are the perspectives from which uncertainty is considered. A system can have many viewpoints, facets and uncertainties. Each uncertainty is characterised by a set of attributes. These attributes have been identified from the literature sources identified earlier, consolidated and extended. The remaining entities of the ERD, from *Risk or Opportunity* to *Outcome*, define the consequences of uncertainties.

Table I shows the uncertainty attributes and associated original domains, and indicates whether they are extended in our framework. Of the twenty seven (27) proposed attributes, all but one can apply to both epistemic and aleatory uncertainties. The attribute *Evidence* is specific to epistemic uncertainty since it depends on the availability of specific knowledge.

Each attribute has an annotation of a superscript denoting its cardinality constraint: $+$, $*$, n or $n+$, such that $+$ means at least one, $*$ means zero or more, n is a positive integer and $n+$ means at least n.

Each uncertainty can result in multiple risks or opportunities. These in turn require one or more mitigations or exploitations with the goal of achieving specific outcomes. The main entities of the framework

[3]https://digital-library.theiet.org/
[4]https://apps.webofknowledge.com
[5]https://dblp.org/
[6]https://www.scopus.com
[7]https://dl.acm.org/
[8]https://ieeexplore.ieee.org

TABLE I
Consolidated uncertainty characterisation attributes.
CgS - cognitive science, CxS - complex systems, CPS - cyber-physical systems, SAS - self-adaptive systems, REL - the RELAX language

Attributes	Options	CgS	CxS	CPS	SAS	REL	Extended
Uncertainty description attributes							
Description[+]	[What]	✓	✓	✓	✓	✓	No
Nature[1]	Aleatory or Epistemic	✓	✓	✓	✓		No
Bound[*]	Known limits around the uncertainty					✓	Yes
Perspective[1]	Objective or Subjective	✓		✓			No
Awareness[1]	Known unknown or Unknown unknown	✓	✓	✓			No
Level[1]	Certainty to uncertainty		✓	✓	✓		No
Uncertainty source attributes							
Source type[+1]	Endogenous or Exogenous		✓				No
Cause[+1]	[Why]	✓	✓	✓	✓	✓	No
Uncertainty system attributes							
Viewpoint[+1]			✓	✓	✓		Yes
Facet[+1]		✓	✓	✓	✓	✓	Yes
Location[+1]	[Where]			✓	✓		No
Uncertainty manifestation attributes							
Manifestation[+1]	Observable impact	✓	✓	✓	✓	✓	No
Measure[+1]		✓		✓	✓		No
Monitor[*]						✓	Yes
Evidence[*]				✓			No
Relationship[+1]	Between evidence and monitor			✓			Yes
Uncertainty time attributes							
Emerging time[+1]	[When]		✓	✓	✓		Yes
Lifetime[+1]	Limited or Perpetuity			✓	✓		No
Change[+1]	Dynamic or Static		✓	✓			No
Pattern[*]	Systematic - periodic, Systematic - persistence, Aperiodic - transient, or Aperiodic - Sporadic			✓			No
Uncertainty mapping attributes							
Dependencies[*]	with other uncertainties of the system					✓	No
Risk[+1] or Opportunity[+1]	Negative or positive consequences of uncertainty		✓				No
Mitigation[+1] or Exploitation[+1]			✓				No
Operator[+1]	Modal, ordinal and temporal operators to express managed uncertainty outcomes					✓	Yes
Outcome[+1]	Uncertainty management objectives		✓				No

are therefore, the system, its facets, viewpoints, risks, opportunities, mitigations, exploitations and outcomes.

A comprehensive consideration of uncertainty requires details of these concepts as well as their relationships to be defined. We propose a generic foundation and do not prescribe implementation details. However, there is some existing work relating to these entities. For instance, viewpoints are often used in the context of software architectures [15].

The software architecture of a system can be represented in terms of views, such as the 4+1 viewmodel [15]. Uncertainties can be incorporated in these views using the framework attributes of Table I and the relationships in Figure 1. Where necessary, additional attributes may be added or removed to suit the context.

The rest of this section describes the uncertainty attributes in the framework, organised into six categories, as shown in Table I: description, source, system, manifestation, time and mapping.

- Uncertainty description attributes:
  - *Description* captures the specifics of the uncertainty in natural or structured language.
  - *Nature* of the uncertainty is either epistemic or aleatory.
  - *Bounds* are the known limits or scope of uncertainty as it occurs.
  - *Perspective* is subjective if the uncertainty depends on the agent defining it, and objective if the uncertainty is absolute.
  - *Awareness* is an agent's consciousness of uncertainty: known unknown or unknown unknown.

  - *Level* is the position of uncertainty in a spectrum from certainty to total uncertainty and can be expressed in different ways, for instance, using fuzzy concepts such as High, Medium and Low.
- Uncertainty source attributes:
  - *Source type* of uncertainty can be internal (endogenous) or external (exogenous).
  - *Cause* is the trigger of the uncertainty, such as humans in the loop, operational interference and business changes.
- Uncertainty system attributes:
  - A *software system* might be composed of subsystems and their components.
  - *Viewpoint* is a standpoint from which we express and analyse uncertainty.
  - *Facet* is a specific aspect of a software system such as activities, artefacts and infrastructure.
  - *Location* is where the uncertainty appears in the system or its facets.
- Uncertainty manifestation attributes:
  - *Manifestation* represents how the uncertainty appears or emerges. Possible options include system states, nature of input, availability of resources or future outcomes.
  - *Measure* is the expression of the degree of uncertainty. Measures might include probability and fuzzy values.
  - *Monitors* detect uncertainty manifestation in the sources of uncertainty. The output of monitors contributes to evidence.
  - *Evidence* demonstrates the existence of uncertainty. Additional evidence, such as objective facts, can reduce epistemic uncertainty.
  - *Relationship* is an intermediary function which can be used to confirm uncertainty manifestation in the absence of direct feedback of detecting uncertainty from monitors.
- Uncertainty time attributes:
  - *Emerging time* is when the uncertainty manifests during the system lifecycle, such as development time or run-time.
  - *Lifetime* indicates whether the uncertainty has a limited lifetime or exists in perpetuity.
  - *Change* indicates how uncertainty is influenced by time, determining whether it is dynamic or static. For instance, epistemic uncertainty can disappear with evidence as knowledge improves.
  - *Pattern* represents the behaviour of uncertainty with time, such as systematic or aperiodic or follows another custom pattern.
- Uncertainty mapping attributes:
  - *Dependency* identifies the uncertainties which are related to the current uncertainty.
  - Uncertainty causes *Risks or Opportunities* which require *Mitigation or Exploitation* to achieve one or more specific *Outcomes*.
  - *Operators* are key words, phrases or terms that can express or signal uncertainty or its constraints. We have used modal, temporal and ordinal categories of operators, as defined in RELAX [16]. In addition, ordinal operators AT LEAST and AT MOST, as used in software architecture facets for traceability [17], [18], and temporal operator AROUND <time> are included in the framework.

A. Applying the Framework

This section outlines one possible workflow which can be used to apply the framework in practice. This workflow is not mandated and others may be used depending on context and need.

Firstly, we identify the system in which uncertainties are considered. Secondly, we identify the set of facets and viewpoints in which uncertainty needs to be incorporated. Thirdly, we identify specific uncertainties in each facet under consideration. Software engineers can identify uncertainties using approaches such as scenarios, brainstorming, experience reports, user feedback, budget details, reference architectures and risk assessment reports. Fourthly, for each uncertainty identified, we record attribute values.

The framework can be used in plan-driven or agile development contexts, as uncertainties are not assumed to be static or required to be fully documented at the outset. As artefacts evolve, the uncertainties associated with them can be updated.

VI. Workbench

The workbench, introduced in Section IV, is implemented by enhancing a clone of an existing open source diagramming software tool, diagrams.net (previously known as draw.io) [9], with support for the framework. The tool can be used to capture uncertainty details in multiple views of the software architecture of a system.

Since diagrams.net supports the encoding of architecture diagrams in data formats such as XML and JSON, the architecture models and the associated uncertainty data can be exported for further processing in other tools or be recorded as part of the architecture knowledge for uncertainty management activities. As part of future work, we plan to extend the workbench to incorporate analysis tools. Since the workbench is

[9]https://github.com/jgraph/drawio

TABLE II
Logical view: Capture component uncertainties

Attribute	Uncertainties			Status
Description	Future data sources	Data importation capacity	Connections to future sources	Explicit
Nature	Epistemic	Epistemic	Epistemic	Inferred
Bound				
Perspective	Objective	Objective	Subjective	Inferred
Awareness	Known unknown	Known unknown	Known unknown	Inferred
Level	High	Medium	Medium	Inferred
Source	Exogenous	Exogenous	Endogenous	Explicit
Cause	Identification of new data source	Data volume growth from sources	Non-standard data sources formats	Inferred
Viewpoints	Logical	Logical	Logical	Explicit
Facets	Architecture	Architecture	Architecture	Explicit
Location	Capture component	Transport component	Transport component	Explicit
Manifestation	New data sources	Data import delays	Failure to connect	Inferred
Measure	Probability	Importation fluctuation		Inferred
Monitor	Governance - data integration requests	Data importation rate	Integration failures reports	Explicit
Evidence	New data source	Data importation history/trend	Data sources connection failure	Inferred
Relationship		Data importation congestion	Number outstanding connections	Inferred
Emerging time	Run-time	Run-time	Development, & Run-time	Inferred
Lifetime	Perpetuity	Perpetuity	Perpetuity	Inferred
Change	Dynamic	Dynamic	Dynamic	Inferred
Pattern	Aperiodic-sporadic	Aperiodic-sporadic	Systematic	Inferred
Dependencies	Transport & Refine components	Refine components	Refine components	Explicit
Risk or Opportunities	Risk: Data Integration failure	Risk: Data loss & import delays	Risk: Data import delay and failure	Inferred
Mitigation or Exploitation	Mitigation: Extensible integrations	Mitigation: Extensible design	Mitigation: Extensibility	Explicit
Outcome (with operators)	Integration with AS MANY future sources AS POSSIBLE	Import AS MUCH data as POSSIBLE	MINIMISE customisation required to connect to new data sources	Inferred

open source and extensible, users may add tools as required thus facilitating adoption in practice [19].

In addition to recording uncertainty data, users can calculate metrics such as the number of uncertainties per architecture component or the number of uncertainties with a specific attribute value in the current view in the workbench.

VII. Case study: Application of the framework

In this section, we apply the framework to the architecture of the Information Management Platform for Data Analytics and Aggregation (IMPALA)[1] used by the National Aeronautics and Space Administration (NASA) as a Big Data reservoir for medical and health data of astronauts to produce data on uncertainties present at this level. We identify the potential uncertainties in the architecture of IMPALA by reviewing its System Design Document (SDD)[1], which describes its requirements, system design and data design.

The IMPALA architecture description has the following views - Logical, Functional, Infrastructure, Network and Security. Each viewpoint is described in the SDD using a diagram showing the relevant architecture elements from the perspective of the viewpoint. In this paper, we will focus on some of the uncertainties in the logical view.

The architecture has seven main components: *Capture, Transport, Refine, Store, Analyse, Distribute and Manage*. Each of these components has its associated uncertainties. Table II shows some of the uncertainties identified for the Capture, Transport, Refine and Store components from their description in the SDD and characterised using the framework. The values of some of the framework attributes are explicitly given in the documentation, while others were inferred from the description of the architecture design. The final column of Table II shows this status.

VIII. Discussion

Table II shows three of the uncertainties in the logical view of the IMPALA architecture. Others are not shown here due to lack of space. We focus on these three uncertainties to illustrate the kind of information we can generate and utilise by applying the framework to software architecture.

In the first instance, values of attributes can be analysed. For example, the *nature* attribute tells us that two uncertainties are aleatory while the rest are epistemic. Since epistemic uncertainty is due to lack of knowledge, we infer that reducing the knowledge gap, if possible, can mitigate the uncertainty. In contrast, an aleatory uncertainty is due to inherent variability, and therefore, other related attributes need to be carefully considered as part of its mitigation.

Similarly, the *location* attribute indicates that there are two uncertainties each in the *Transport* and *Store* components. *Capture* and *Refine* components have one each. We can use such information as a heuristic to predict the vulnerability of an architecture component to uncertainty. A range of useful uncertainty metrics from raw attribute values to aggregated or derived figures can be defined.

A consideration of possible mitigations in Table II shows that managing uncertainties can impact architecture design. For example, the use of extensibility in

design relates to fundamental architecture decisions. Candidate architectures can be compared or evaluated based on their uncertainties and potential mitigations to achieve desired outcomes.

Although risks from localised uncertainties are mitigated in the context of their components, this architecture design is still vulnerable to uncertainties from major component failure. For instance, if the *Transport* component fails due to a prolonged network outage, the whole IMPALA platform risks failure as this is a bottleneck component without a backup alternative. The framework will allow such uncertainties to be recorded in appropriate views of the architecture and linked to allow an assessment of their impact.

While there are benefits to using the framework and a supporting toolset, there are also challenges in its application. For example, in order to facilitate automated analysis and calculation of metrics, the types of values of framework attributes may need to be constrained. While the framework allows users to consider known uncertainties in a systematic manner, it is possible that unanticipated uncertainties or those without clear mitigations are missed.

At a broader level, software professionals may be concerned with the robustness of systems, which requires the management of uncertainties such as the ones discussed in this paper as well as errors in inputs, data, software and hardware used by the system.

IX. Conclusions and future work

This paper has outlined a foundational framework for considering uncertainty in software systems, and demonstrated the application of the framework to characterise and document uncertainties in a software architecture model. The immediate benefit of this work is the definition of an explicit and concise approach to capturing and representing uncertainty knowledge in software engineering. As future work, we aim to support uncertainty knowledge management and analysis in multiple artefacts and further explore the underlying predictive and analytic application of this knowledge.

Acknowledgment

This work is partly funded through a PhD studentship to the first author by the School of Computer Science, University of St Andrews.

References

[1] C. Lupafya, "A framework for managing uncertainty in software architecture," in *Proceedings of the 13th European Conference on Software Architecture-Volume 2*, pp. 71–74, ACM, 2019.

[2] R. Kazman, M. Klein, and P. Clements, "Atam: Method for architecture evaluation," tech. rep., Carnegie-Mellon Univ Pittsburgh PA Software Engineering Inst, 2000.

[3] M. Smithson, *Ignorance and uncertainty: emerging paradigms*. Springer Science & Business Media, 1989.

[4] W. E. Walker, P. Harremoës, J. Rotmans, J. P. Van Der Sluijs, M. B. A. Van Asselt, P. Janssen, and M. P. von Krauss, "Defining uncertainty: a conceptual basis for uncertainty management in model-based decision support," *Integrated assessment*, vol. 4, no. 1, pp. 5–17, 2003.

[5] N. Esfahani and S. Malek, "Uncertainty in self-adaptive software systems," in *Software Engineering for Self-Adaptive Systems II*, pp. 214–238, Springer, 2013.

[6] D. Perez-Palacin and R. Mirandola, "Uncertainties in the modeling of self-adaptive systems: a taxonomy and an example of availability evaluation," in *Proceedings of the 5th ACM/SPEC international conference on Performance engineering*, pp. 3–14, ACM, 2014.

[7] A. J. Ramirez, A. C. Jensen, and B. H. C. Cheng, "A taxonomy of uncertainty for dynamically adaptive systems," in *Proceedings of the 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pp. 99–108, IEEE Press, 2012.

[8] S. Mahdavi-Hezavehi, P. Avgeriou, and D. Weyns, "A classification framework of uncertainty in architecture-based self-adaptive systems with multiple quality requirements," in *Managing Trade-Offs in Adaptable Software Architectures*, pp. 45–77, Elsevier, 2017.

[9] D. Hastings and H. McManus, "A framework for understanding uncertainty and its mitigation and exploitation in complex systems," in *2004 Engineering Systems Symposium*, pp. 29–31, 2004.

[10] O. L. de Weck, C. Eckert, and J. Clarkson, "A classification of uncertainty for early product and system design," in *International Conference on Engineering Design*, Massachusetts Institute of Technology. Engineering Systems Division, 2007.

[11] M. Zhang, B. Selic, S. Ali, T. Yue, O. Okariz, and R. Norgren, "Understanding uncertainty in cyber-physical systems: a conceptual model," in *European conference on modelling foundations and applications*, pp. 247–264, Springer, 2016.

[12] M. Zhang, S. Ali, T. Yue, R. Norgren, and O. Okariz, "Uncertainty-wise cyber-physical system test modeling," *Software & Systems Modeling*, vol. 18, no. 2, pp. 1379–1418, 2019.

[13] N. Esfahani, "A framework for managing uncertainty in self-adaptive software systems," in *Proceedings of the 2011 26th IEEE/ACM International Conference on Automated Software Engineering*, pp. 646–650, IEEE Computer Society, 2011.

[14] D. Garlan, S.-W. Cheng, A.-C. Huang, B. Schmerl, and P. Steenkiste, "Rainbow: Architecture-based self-adaptation with reusable infrastructure," *Computer*, vol. 37, no. 10, pp. 46–54, 2004.

[15] P. Kruchten, "Architectural blueprints–the" 4+ 1" view model of software architecture," *IEEE Software*, vol. 12, no. November, pp. 42–50, 1995.

[16] J. Whittle, P. Sawyer, N. Bencomo, B. H. Chengy, J. M. Bruelz, B. H. C. Cheng, and J.-M. Bruel, "RELAX: Incorporating uncertainty into the specification of self-adaptive systems," *Proceedings of the IEEE International Conference on Requirements Engineering*, pp. 79–88, 2009.

[17] A. Ghabi and A. Egyed, "Exploiting traceability uncertainty between architectural models and code," in *Proceedings of the 2012 Joint Working Conference on Software Architecture and 6th European Conference on Software Architecture, WICSA/ECSA 2012*, 2012.

[18] C. Trubiani, A. Ghabi, and A. Egyed, "Exploiting traceability uncertainty between software architectural models and performance analysis results," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9278, 2015.

[19] I. Malavolta, P. Lago, H. Muccini, P. Pelliccione, and A. Tang, "What industry needs from architectural languages: A survey," *IEEE Transactions on Software Engineering*, vol. 39, no. 6, pp. 869–891, 2013.