JOT
JOURNAL OF
OBJECT TECHNOLOGY

Interface Contracts for Workflow+ Models: an Analysis of Uncertainty across Models

Richard F. Paige*, Fiona A.C. Polack†, Steffen Zschaler‡, Thomas Chiang*, and Nicholas Annable*
*McMaster University, Canada
†University of Hull, United Kingdom
‡King's College London, United Kingdom

ABSTRACT Workflow models are used to rigorously specify and reason about diverse types of processes. The Workflow+ (WF^+) framework has been developed to support unified modelling of the control and data in processes that can be used to derive assurance cases that support certification. However, WF^+ is limited in its support for precise contracts on workflow models, which can enable powerful forms of static analysis and reasoning. In this paper we propose a mechanism for adding interface contracts to WF^+ models, which can thereafter be applied to tracing and reasoning about the uncertainty that arises when combining heterogeneous models. We specifically explore this in terms of design models and assurance case models. We argue that some of the key issues in managing some types of uncertainty can be partly addressed by use of interface contracts.

KEYWORDS Uncertainty, modelling, process models, interface contracts.

1. Introduction

A model captures information about phenomena of interest, e.g., a system's properties, software's behaviour, constraints on a design. Very often, models are developed iteratively and incrementally, in response to improved understanding about said phenomena. This improved understanding can be acquired through user-centred means (e.g., interviews with stakeholders, design workshops), through analysis of the incomplete or partial models that have been so far derived, or through application of refinement and refactoring rules that allow engineers to systematically improve models.

Of the many different types of models used in systems engineering, for architecture and design, requirements (e.g., use cases, SysML requirements diagrams), etc., this paper focuses on process models. A process model captures a set of activities or tasks that can be carried out in some order, according to some schedule, by a set of actors. A number of process modelling languages have been developed, including business process modelling languages (the current standard is Business Process Definition Metamodel[1]) and activity diagrams.

Going beyond the foundations of process modelling, Workflow+ (WF^+) (Annable et al. 2022) is a framework that has been developed to provide unified modelling of data and control in processes. Originally developed to help specify safety-related processes in critical systems engineering, WF^+ provides a set of modelling concepts for capturing and reasoning about the interactions between data and control in a variety of processes. As a framework, it needs to be instantiated in order to provide a domain-specific modelling language for a particular context (e.g., modelling aerospace software engineering processes). WF^+ can be used to document work already done, as well as processes that remain to be enacted.

A key goal of WF^+ is to model processes in a way that supplies information needed for (safety) *assurance*. In particular, it was developed to help systematise the generation of *assurance cases*, with support for traceability between data and control activities and the derived assurance case elements. This in turn enables other forms of analysis, especially impact analysis. These elements have been explored in depth in a recent project

JOT reference format:
Richard F. Paige, Fiona A.C. Polack, Steffen Zschaler, Thomas Chiang, and Nicholas Annable. *Interface Contracts for Workflow+ Models: an Analysis of Uncertainty across Models*. Journal of Object Technology. Vol. 21, No. 4, 2022. http://dx.doi.org/10.5381/jot.2022.21.4.a6

[1] https://www.omg.org/spec/BPDM

with an industrial partner in the safety-critical domain.

A significant limitation of WF$^+$, discovered during the aforementioned project, is its support for modelling the activities that make up behaviours: they are modelled as coarse-grained artefacts, with no way to represent what each activity *requires* in order to successfully complete its tasks, or what successful completion then *ensures*. This lack of a notion of an *interface contract* limits the scope and depth of impact analysis, and, indeed, other forms of analysis.

Interface contracts are particularly important with respect to analysing and reasoning about *uncertainty* with respect to a process model. Interface contracts express constraints on process models so as to *highlight* uncertainty, *constrain* activities so that uncertainty is more manageable, and define how processes can be connected with other processes to mitigate or eliminate uncertainty: this is a notion of inter-modelling, i.e., composing two (or more) models at potentially different levels of abstraction. To illustrate, consider the situation where there are two process models expressed in UML activity diagrams, one models a software engineering process used by an organisation, while the other models a safety engineering process. It is desired to inter-relate these two models via trace links, for example, connecting a *Review* activity in the software engineering process, with a *Hazard Analysis* activity in the safety engineering process. Consider as well the situation where the software engineering activity of *Review* is subject to uncertainty: the process model does not capture what constitutes a Review, what inputs it requires, what it produces, and what it depends on. We argue that providing interface contracts on process models can help manage this situation, and others.

This paper makes several contributions. It summarises the WF$^+$ framework for modelling, and describes its Model-Driven Engineering (MDE) underpinnings. The limitations of WF$^+$ are also highlighted. We then present a lightweight extension to WF$^+$ to support interface contracts, particularly preconditions and postconditions. We illustrate both the original WF$^+$ and its extension with a representative example from the safety engineering domain. Finally, we highlight opportunities that arise from this extension in terms of supporting richer forms of analysis, particularly in supporting forms of analysis with respect to uncertainty.

Our motivation for writing this paper is twofold: to propose an extension to WF$^+$ to address the specific process modelling challenge of supporting richer forms of analysis; and to illustrate challenges pertaining to uncertainty modelling in the context of process models.

2. Related Work

2.1. Process modelling

Process modelling is the description of activities and their interrelationships (Rolland et al. 1999). A *process model* captures an abstraction of a workflow of some kind. Process modelling is broadly applied in systems and software engineering. Particular applications include *business process modelling* (Laguna & Marklund 2004) which captures activities and interrelationships pertaining to the behaviour of an enterprise, and *software process modelling*, which captures software engineering activities and their interrelationships. In the former, a wide range of business processes have been captured in real-world examples, including procurement processes, manufacturing, insurance claim processing, and others. In the latter, a range of software engineering processes (e.g., waterfall, spiral, V-model) have been modelled in order to support scheduling, critical path analysis, workload modelling, etc. The purpose of capturing these models is multifold: to enhance understanding of what may be sophisticated and complicated processes; to enable analysis, e.g., to identify bottlenecks or sources of delay in process; and to support automation, e.g., to automate the management and execution of workflows, including determining to what extent a process has executed.

Numerous languages have been proposed to support process modelling across a range of domains. UML activity diagrams (*UML 2.5.1 Specification* 2017) provide a domain-agnostic syntax and semantics that supports capturing activities, dependencies between activities, synchronisation points, and concurrency. Activity diagrams are a rich formalism supported by numerous modelling tools (e.g., Papyrus, Visual Paradigm, PTC Integrity Modeller), and have been used for many purposes, including simulation and workflow modelling. However, because they are founded on UML state diagrams, they do not make it immediately clear which objects in a system execute which activities. It is difficult to express hierarchy in activity diagram models of workflows: a new activity diagram is required for each workflow. As a result, diagrammatic models become complicated very quickly.

Business Process Modeling and Notation (BPMN) (*BPMN 2.0 Handbook Second Edition: Methods, Concepts, Case Studies and Standards in Business Process Management Notation* 2011) shares some features with activity diagrams (e.g., swimlanes and flow) but introduces concepts specifically for business process modelling, including gateways and transactions. It has seen substantial use in domains such as automating procurement processes, insurance claim handling processes, and supporting low-code development. BPMN and activity diagrams have very similar expressiveness. However, BPMN does not support data modelling, nor does it provide the level of formality that activity diagrams does through its metamodel and its static semantic constraints. BPMN does have mappings to an implementation platform, BPEL, though this mapping is not always straightforward.

The process modelling approaches discussed above focus on modelling processes that are to be executed—aiming, *inter alia,* to ensure that they are executed correctly or to provide (partial) automation of process execution. There is an alternative perspective: often it is important to *document* processes that have happened. This is, for example, important when constructing safety assurance cases. More generally, documentation of the processes that have led to the development of an artefact has been studied using the notion of data provenance. Generic and fundamental concepts for data provenance have, for example, been captured and standardised in the PROV ontology (Lebo et al. 2014).

From a process modelling perspective, work on WF$^+$ has so

far focused on how to capture information relevant to safety in WF$^+$ metamodels so that when it is instantiated we can support assurance case development. Ultimately the framework aims to enable the use of WF$^+$ metamodels as reusable templates that guide development of safety-critical systems and their assurance, but we have not yet considered how to model how the workflow should be executed. One benefit of adding contracts is that we take a step towards specifying how a workflow should be executed by enabling conditions on when a process can begin and be considered completed.

2.2. Uncertainty modelling

Uncertainty is a pervasive issue in software and systems engineering. There are different sources of uncertainty, such as uncertainty about measurements or uncertainty in predictions of system properties from early-stage design models. However, irrespective of the source of uncertainty, it is important that as many as possible of the known uncertainties are captured explicitly to enable a systematic and well-founded development process. This is particularly the case for system properties about which an assurance case must be made, as it is important to understand the robustness of the assurance argument.

Uncertainty is a topic of increasing interest in many areas of computer science, building on attempts to define a useful classification of uncertainty. Some examples of classifications based on review of existing ontologies are

- Jousselme et al. (Jousselme et al. 2003): a classification from a situational analysis perspective, distinguishing, for example, ignorance (a state of mind) and uncertainty (a consequence of limitations in observation);
- Padulo and Guenov (Padulo & Guenov 2012): a separation into uncertainty about and uncertainty within a design space or problem.

A useful classification comes from Esfahani and Malek (Esfahani & Malek 2013), who identify two uncertainty dichotomies:

- *reducible* versus *irreducible*, which underpins management of uncertainty;
- *aleatory* versus *epistemic*, which capture the essential nature of an uncertainty, where aleatory uncertainty is a variation in the underlying system (for example, due to apparently random events – this form of uncertainty is often represented by statistical prediction models or probabilities) and epistemic uncertainty is due to a lack of knowledge (for example about a parameter or process in the system, which can be resolved).

In (Bernardi et al. 2021), irreducible uncertainty is elaborated as phenomena that are inherently unknowable – uncertainty that persists even in the presence of complete information. Reducible uncertainty can, however, be resolved by improved understanding. Irreducible uncertainties need to be recorded: they are, in a sense, facts about the limitation of certainty.

Reducible aleatory uncertainty cannot be eliminated, but models used to derive distributions or probabilities may be improved: the recording of what the uncertainty appears to be and how it is modelled is important, but quantification of the effect can never be exact. Reducible epistemic uncertainty is subject to cost-benefit: some can eventually be eliminated (e.g. because a parameter or process is tied down in implementation), but it may not be worth the effort.

The most comprehensive review of uncertainty in model-based engineering is the systematic review conducted by Troya et al. (Troya et al. 2021). The paper brings together academic and practically-based work addressing modelling uncertainty, under the headings of behaviour, belief, design, measurement, occurrence, and spatiotemporal. The authors note the immaturity of most work on uncertainty, but also the growing body of work on real and industrial problems. None of the studies surveyed looks at orthogonal models such as the assurance process (model). In considering workflows and assurance, we are dealing with at least behaviour, belief, measurement and occurrence. Of more than 200 papers included in the survey, a minority address real-world (including critical systems) examples, but none addresses assurance (Troya et al. 2021).

There is a wide range of research on modelling and managing uncertainty. Vallecillo (Vallecillo 2019), Zhang et al. (Zhang et al. 2016), and others, propose metamodel extensions to record belief. Belief recording is useful, but does not currently support evaluation of cumulative uncertainty or its effect. Attempts to quantify uncertainty or express the limits of certainty have focused on statistics and probability, for instance through Bayesian systems (Dai et al. 2007), or mathematical modelling of uncertain characteristics, e.g. (Oberkampf et al. 2002; Patelli et al. 2012): these approaches require some insight into the distribution of possible values (e.g. of a parameter, or the strength of a belief), and are themselves subject to (undocumented) uncertainty. Fuzzy logic is also unable to handle subjective uncertainty, but, more recently, subjective logic admits subjective levels of uncertainty, with some scope to combine uncertainties e.g. (Munoz et al. 2021).

In this paper, we use a simple classification, differentiating only between aleatory and epistemic uncertainty.

3. Workflow+: an example of process modelling

In this section we provide an overview of the WF$^+$ framework, introduce a running example and highlight some key limitations of WF$^+$.

WF$^+$ provides a unified view of data and control for modelling of processes (Annable et al. 2022). This is particularly useful where activities must be carried out in order to ensure safety. This has been used in practice to model industrial safety assurance activities and can also model the software development process; more details on WF$^+$ can be found in (Annable 2020), with an overview of developing tool support in (Chiang 2021).

The WF$^+$ framework requires experts in specific domains (such as automotive or aerospace) to encode their knowledge on what must be done during development to support safety cases. Knowledge is encoded in the form of a *WF$^+$ metamodel*. A WF$^+$ metamodel is a domain-specific metamodel that is defined using the WF$^+$ domain-specific metamodelling

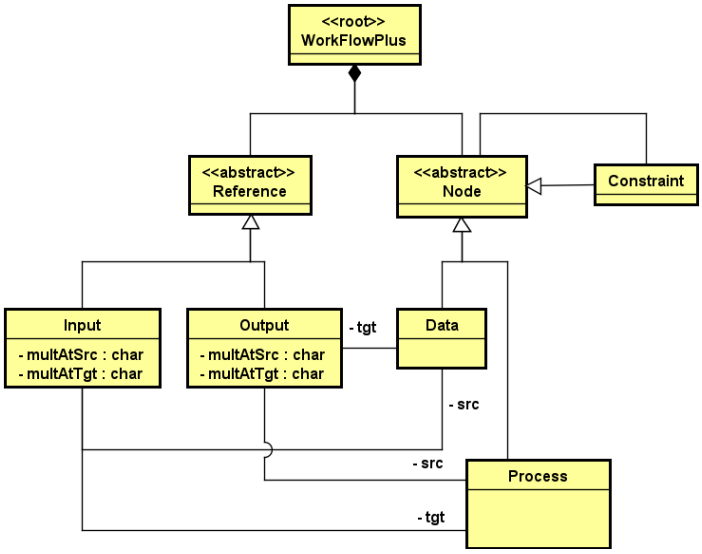

Figure 1 A subset of the metamodel of WF$^+$.

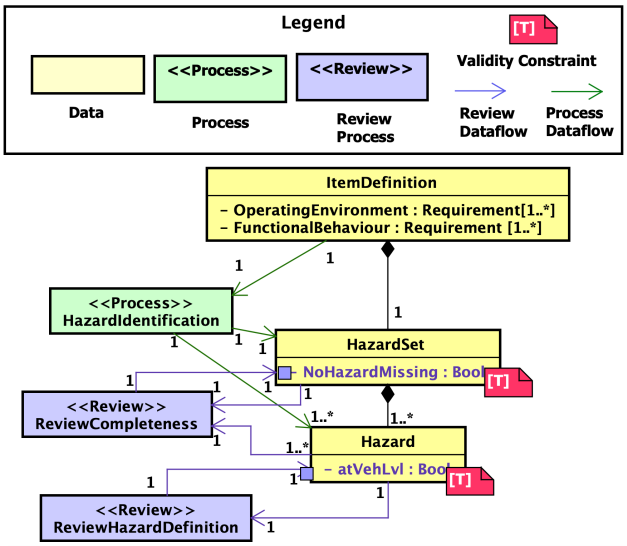

Figure 2 WF$^+$ representation of the Hazard Identification process.

language (Zschaler et al. 2010) shown in Figure 1; that is, it defines a modelling language for specific types of processes. As such, it must be instantiated to provide a specific modelling language for a specific problem.

When the workflow defined in a WF$^+$ metamodel such as that in Figure 2 is executed, the metamodel is instantiated to document that realization. Constraints included on the metamodel are designed to ensure that activities are conducted so as to satisfy the intent of the workflow; the constraints form the basis of derived (safety) argumentation about what the realization of the process guarantees.

3.1. Safety Assurance Example

Our running example uses models based on ISO standard 26262, an automotive functional safety standard (ISO 26262 2018), specifically parts of the Hazard Analysis and Risk Assessment (HARA) defined in ISO 26262 Part 3. The models are focused for this paper and simplified to improve readability. These models are referenced in later sections to illustrate sources of uncertainty in WF$^+$ models. Figure 2 in particular is used to illustrate interface contracts in WF$^+$.

Figures 2 and 3 show WF$^+$ metamodels capturing part of the requirements defined for HARA. We use green classes to represent *processes*, purple classes to represent *generic processes* such as reviews, and yellow classes to represent *product data*. The difference between processes and generic processes is beyond the scope of this example, but we maintain the distinction since they play different roles in the model. Directed arrows show data flow. Purple text with "ports" are used as a shorthand notation for attributes produced by reviews. Multiplicities on the process end of data flow arrows constrain the number of executions that an instance is used by; those on the data (yellow) end specify the number of instances of data used or produced

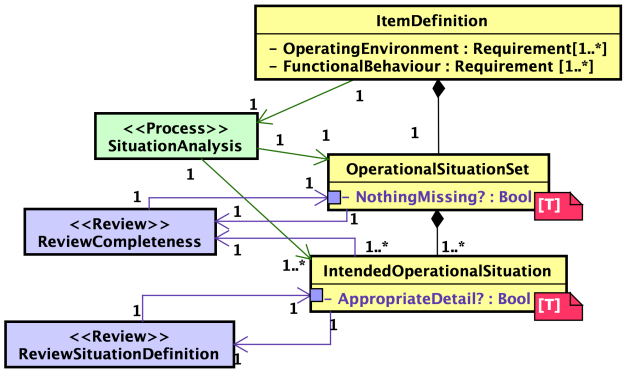

Figure 3 WF$^+$ representation of the Situation Analysis Process.

per execution[2]. Figures 2 and 3 also show WF$^+$ *validity constraints*, using red model annotations: for example, the validity constraint on `HazardSet.NoHazardMissing` is that this must be True in an instance for the consequent safety assurance to hold. Validity constraints specifically apply to the value of attributes produced by reviews, as opposed to regular structural constraints such as the multiplicities shown. We make this differentiation because of the important role external validation plays in safety assurance.

Figure 2 is a simplification of the Hazard Identification WF$^+$ model presented in (Annable et al. 2022). The specific (green) process defined is Hazard Identification. The Item Definition in this simplified example, which is provided by an external development process, can be thought of as a set of requirements for a system or component that is to be part of a vehicle. The purpose of this process is to identify the set of Hazards the system may give rise to based on these requirements. The identified hazards

[2] Note that the full WF$^+$ model also expresses dataflow from a process to association in the output data and between input and output data, but these are omitted here for readability.

are subject to two generic (purple) Review processes that must be realized for safety assurance. Each Review process sets the value of data attributes (`NoHazardMissing`, `atVehLvl`: it is common for such data attributes to be Booleans). Figure 3 is very similar, but defines the specific process, Situational Analysis, and its associated data; the Review processes pertain to the applicability and completeness of the identified intended operational situations. An example of an instance of a WF$^+$ metamodel can be found in (Annable et al. 2022).

3.2. Limitations of WF$^+$

The WF$^+$ framework currently provides constructs to define workflows as WF$^+$ metamodels and to generate (safety) argumentation based on constraints in those metamodels. This generation is demonstrated in (Annable et al. 2022). Prototype tooling supports creation of instances documenting realizations of the workflow, checking against constraints and checking if arguments based on those constraints hold. There are several limitations of the current framework, including ways to link versions of the same process model (e.g. a coarse grained hazard analysis process with a more detailed version of the same process). There is also currently no support for ports, or for interfacing between processes. Data flow arrows show which processes use or produce data, which implies dependency, but WF$^+$ does not explicitly model process dependency.

The limitation addressed in this paper is the need for contracts in WF$^+$ models. Aside from checking the structure of input, there is currently no way of specifying when a process can legitimately be executed, or when it can legitimately be considered finished. This also introduces the issue of uncertainty in processes/assurance. For example, in Figure 2, if there is uncertainty in the requirements, the Hazard Identification process might not be triggered. Similarly, if the Hazard Identification and Situation Analysis processes have been executed but not reviewed we may not want to trigger the Determine Relevant Combinations process, even though all the required data is present in the model. In the following, we introduce interface contracts, and discuss briefly the effects of uncertainty.

4. Interface contracts for WF$^+$

In this section we present a lightweight extension to the WF$^+$ framework to support interface contracts. By adding these contracts, we allow for the precise specification of constraints that otherwise could not be imposed on a WF$^+$ workflow.

The WF$^+$ constraints discussed previously are intended to be evaluated statically on a workflow that has been completely realized so they can be used as part of evidence in an assurance case. In contrast, constraints in interface contracts can be evaluated dynamically during the process of realizing a workflow. This provides a way to more precisely guide the realization of a workflow and in some cases reduce uncertainty in the resulting models.

The key extensions to the metamodel are the addition of Contract, Precondition and Postcondition as shown in Figure 4. A Contract is a special kind of node, one that is associated with a Process. A Process may be associated with an arbitrary

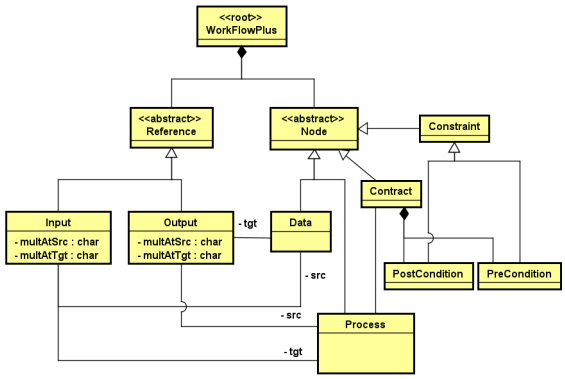

Figure 4 Metamodel of WF$^+$ with the addition of interface contracts.

number of contracts (this could, for example, allow support for versioning of contracts). Each Contract is thereafter composed of a Precondition and a Postcondition, which are themselves (OCL) constraints.

Figure 5 shows an example of contracts added to the example in Figure 2. Starting with the contract for the Hazard Identification process, we include the precondition, specified using OCL, requiring that the item definition contains functional behaviours before the process can be initiated. While this seems to repeat the "[1..*]" constraint in ItemDefinition, it plays a very different role. While realizing a workflow there will inevitably be constraints that are unsatisfied, but that does not always mean we cannot proceed with executing the workflow. By including this constraint in the precondition, we can precisely specify constraints that must hold before the process may begin. In the event that a constraint in a contract is not satisfied, useful feedback could be provided to practitioners realizing the workflow on what they need, and in an integrated model where they need it from, in order to proceed. The postcondition specifies that when the validity constraint on Hazard Set is satisfied, then the process is considered completed. This requires the validity constraint on Hazard to be satisfied for every instance, since this is a precondition for Review Completeness to be executed.

Since reviews are processes, they can also have contracts associated with them. This is also shown in Figure 5. The lower review, Review Hazard Definition, has no precondition, and its postcondition is simply that the output has been produced. Ultimately we want reviews to output the value corresponding to approval, but during the realization of a workflow a negative result from a review is acceptable as far as the contract is concerned. The upper review, Review Completeness, has a precondition that all Hazards must be reviewed and approved before the review can be initiated. Its postcondition also only requires that an output is produced.

From this example, we can begin to see the potential of contracts to address some of the limitations discussed in Section 3.2. By using contracts to control when the Hazard Identification and Review Completion processes can be executed and when they can be considered as completed, we have a greater ability to model dynamic control flow of processes.

Next, the inclusion of constraints on the output of Review

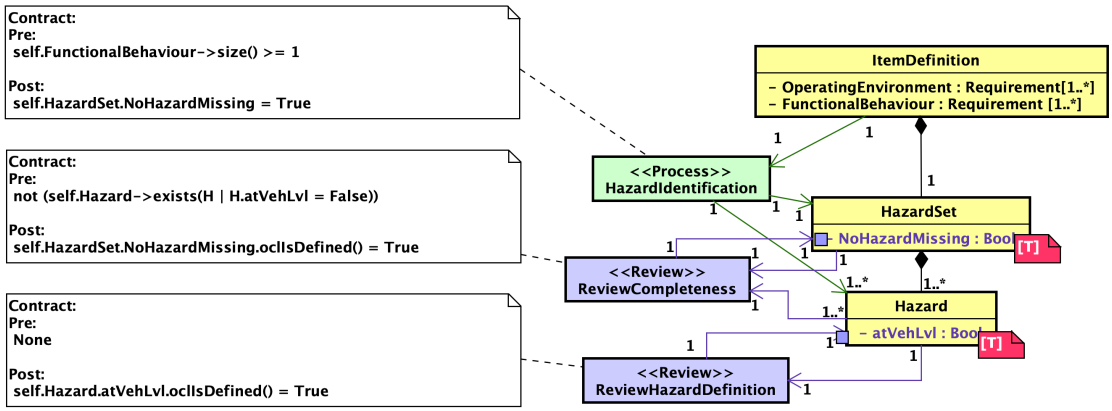

Figure 5 Refined WF$^+$ model of Hazard Identification process from Figure 2 with the addition of interface contracts.

Hazard Definition in the precondition for Review Completeness shows how the dependencies between processes can be strengthened by contracts. Similarly, the reference to FunctionalBehaviour but not OperatingEnvironment in the contract for Hazard Identification illustrates how dependencies between processes and their input data can be strengthened. Both of these benefits are particularly important for impact analysis.

Finally, by adding rigorous interface contracts to processes that define what an execution must satisfy to be acceptable, we enable a design-by-contract approach to workflow design and the inherent information hiding and modularization. Using the contract as a guide, a process could be refined such that it adheres to its contract and maintains compatibility with other processes, but does not reveal secrets about its implementation. Refinement of WF$^+$ models based on their contracts is left for future work.

5. Analysis

Here we briefly outline opportunities for the analysis of uncertainty and impact in WF$^+$ models, and how these are affected by interface contracts. The aim of this section is to identify challenges and opportunities for future research.

5.1. Uncertainty

Analysis of uncertainty requires identification, evaluation (what sort of uncertainty, aleatory or epistemic, what effects might it have on assurance, etc.), and representation / documentation. Approaches to *identification* include *ad hoc* and systematic approaches; the latter are often referred to as deviational analysis, or what-if analysis, and are often derived from the principles of HAZOP. Identification here is quite *ad hoc*, focusing on illustration rather than practical analysis of uncertainty.

Evaluation seeks to understand the nature and potential mitigation of an uncertainty. Here, we discuss the nature and possible effects of a subset of identified uncertainties: this is an area that needs further research, and would explore not only aleatory or epistemic, but also the categories of uncertainty identified in Sect. 2, and might consider for example using subjective logic (Jøsang 2016) to try to put a value on the residual uncertainty in an assurance argument.

Representation could draw on existing uncertainty notations (see Sect. 2), and can also be captured in assurance cases; this is not considered in this paper.

The next section gives examples of aleatory and epistemic uncertainty from specific features of the WF$^+$ model introduced in Sects. 3 and 4. The following section looks at challenges of a more general categorisation of uncertainty in WF$^+$ models.

5.1.1. Identification of uncertainty in data attributes

To illustrate uncertainty in WF$^+$ models, we focus on uncertainty relating to the data attributes: a more complete analysis should focus on each model concept (item, process, data flow etc.) in turn. Revisiting Figs. 2 and 3, we see six specific attributes. We illustrate with one attribute of `ItemDefinition` and the attribute of `HazardSet` and the joint uncertainty of these two attributes.

1. `ItemDefinition.OperatingEnvironment` is modelled as a non-empty set of requirements. Unpicking the modelling, this means that there must be one instance of the operating environment, but there may be any number of operating environments.

   Aleatory uncertainties: there is no way to know whether all aspects of the operating environment(s) of relevance have been specified, or whether any aspect is represented in sufficient detail. Real examples of this type of situation can be found on any site reporting self-drive car accidents, which typically occur because the operating environment included situations that had not been foreseen by the designers / assurance process.

   Epistemic uncertainties: we do not know, but could potentially know, what constitutes an operating environment or how many there are. In this case, the ability to refine or elaborate a WF$^+$ model would allow us to define the operating environment concept more clearly, reducing uncertainty. A key point here is the engineering imperative of cost-benefit: as in areas such as assurance and testing, the engineer needs to determine how much refinement or elaboration is appropriate, trading off further design / modelling against value added to the product or its assurance.

2. `HazardSet.NoHazardMissing: Bool` has its value set by the realisation of the generic review process; it must be *True* for the overall process realisation to be completed (and the relevant assurance case generated). Safety cases almost always argue over a set of things that is asserted to be complete (argue over all hazards, for instance), and this is an inherent source of uncertainty in most safety critical systems evaluation.

   Aleatory uncertainties: there is no way to know whether anything is missing from the hazard set. This uncertainty is mitigated by using techniques such as checklists and brainstorming, but cannot be eliminated (we cannot prove a negative). As in traditional safety case analysis, the assurance is always predicated on the identified hazard set.

   Epistemic uncertainties: again, epistemic uncertainties relate to missing but obtainable information. For instance, there may be identifiable hazards that are not fully described or depend for their detail on aspects of the vehicle design that have not yet been resolved. Interface contracts can make it easier to trace such connections. For example, in Figure 5, the contract for `HazardIdentification` simply states that there must be at least one `FunctionalBehaviour`. This could be refined to require that all `FunctionalBehaviours` must be `resolved` before `HazardSet.NoHazardMissing` can be set to `true`.

3. Although the corresponding WF$^+$ metamodel is not presented in this work, the next step in HARA is to determine relevant combinations of Hazards and Intended Operational Situations to form Hazardous Events. For example, a particular hazard might apply in intended operating situations where the vehicle is moving, but not when it is stopped. Joint uncertainty of the two attributes refers to the effects of uncertainty in the operating environments and a hazard set, as they affect the specification of hazardous events.

   Aleatory uncertainties: if not all operational environments of relevance have been specified, then it is impossible to determine if all intended operating situations in which a hazard is relevant are identified. Since both attribute sets cannot be proved "complete" there will always be irreducible aleatory uncertainty. It may be that a subjective or fuzzy logic, or a belief-based representation can be used here to try to evaluate both the individual attribute uncertainties and the combined uncertainty. However, we cannot remove this type of uncertainty.

   Epistemic uncertainties: here, it is theoretically possible to consider each epistemic uncertainty in the operating environment set in combination with each epistemic uncertainty in the hazard set. In some cases, the uncertainty is reduced or eliminated as the process is refined or elaborated, and in others it may be reduced or eliminated as the system development (e.g. the software engineering, and the design of the vehicle itself) progresses. As for individual attribute uncertainties, other combined epistemic uncertainties may be determined to be not worth resolving, and are simply recorded as limitations on the assurance of the system.

5.1.2. Categorising sources of uncertainty in WF$^+$ models

WF$^+$ models represent software / systems engineering processes and the artefacts (e.g., software models) that they manipulate. Here we identify some general occurrences of uncertainty.

A first potential source of uncertainty is in the artefacts (e.g. models) being manipulated (Bernardi et al. 2021). A second potential source of uncertainty is in the processes that manipulate artefacts. Here most uncertainty arises because processes are undertaken by people, but there may also be uncertainty in process outcomes. Finally, there may be uncertainty in the WF$^+$ models themselves, because there is insufficient knowledge about the real processes.

We illustrate each of these sources, showing examples of uncertainty in each case and discussing opportunities for analysis, with a particular focus on how uncertainty propagates through a workflow of processes based on a WF$^+$ model, possibly with interface contracts. Again, we consider only aleatory vs epistemic uncertainty in this initial discussion.

1. *Uncertainty in models/artefacts.*

   (a) **Aleatory uncertainty.** This is uncertainty that is inherent to the underlying system. For example, measurement uncertainty limit what can be known about the precise value of a property of the system. Similarly, environment uncertainty limits what can be known about the environment in which the system may operate—even though this has potentially substantial impact on system behaviour. An example of this type of uncertainty can be seen in Figure 3, where realizations of `ItemDefinition.OperatingEnvironment` will inevitably have uncertainty since they are an abstraction of an infinitely complex environment.

   (b) **Epistemic uncertainty.** This is uncertainty in the models that reflects insufficient knowledge about the system being modelled. For example, we might not have made a decision between several design alternatives yet or there may be uncertainty about the precise operational temperature range for a particular component. An example of this type of uncertainty can be seen in Figure 2, where realizations of `ItemDefinition.BehaviouralRequirement` used to identify hazards may be incomplete or may not accurately represent the system being developed.

Uncertainty analysis and interface contracts. Uncertainty in models and artefacts may be propagated by processes manipulating these models and artefacts. Where a process uses a model or artefact as its input, the process's outputs may exhibit corresponding uncertainty, as a result. WF$^+$ on its own already provides information about the inputs and outputs of a process and this can be further

refined by an interface contract. Modelling the structure of entities in more detail allows a more fine-granular tracking of uncertainty. Similarly, pre- and post-conditions in interface contracts may provide some indication of how outputs are derived from inputs—further refining the potential for analysis. An example of this can be seen in Figure 5. If there is still uncertainty in the quality of identified Hazards and not all `Hazard.AtVehLvl` are `True`, the contract on `ReviewCompleteness` prevents the review from being triggered and prevents uncertainty in the Hazards from being propagated.

2. *Uncertainty in processes*

   (a) **Aleatory uncertainty.** This is uncertainty inherent to the actual processes. WF^+ is used to capture processes undertaken by humans and this invariably leads to a degree of subjectivity and, therefore, uncertainty in outcomes. An example of this type of uncertainty can be seen in Figure 2, as the process of determining the set of possible hazards is complex and difficult and may have uncertain outcomes. Many techniques used in practice such as comparison with results for similar systems, reviewing by one or more experts and precisely defined guidance aim to reduce uncertainty in process outcomes, but there always remains an irreducible core uncertainty.

   (b) **Epistemic uncertainty.** This is uncertainty in our knowledge about the processes. In other words, it is uncertainty that manifests in the models of the processes—WF^+ models in our example. Some of this uncertainty may be due simply to the abstraction inherent to every modelling process. For example, in WF^+ we largely abstract from the detailed steps that form a process, capturing information about what a process does only at the level of its inputs and outputs and any changes effected on these (the latter captured through interface contracts). In other cases, epistemic uncertainty in processes arises because we do not know enough about the processes. This may be because we haven't understood or designed the process in sufficient detail, but it may also be because of limited information available about the process. The latter can, for example, occur were different organisations or departments collaborate and are hesitant about sharing details of their respective processes. For example, hazard analysis is a widely studied topic, and the Hazard Identification process in Figure 5 could be refined to implement some known technique using the interface contract as guidance to reduce uncertainty. Intelligent modularization of processes using interface contracts can reduce uncertainty in integrating processes when collaborators are hesitant to share information. For example, later processes can know exactly what Hazard Identification will produce without knowing how.

   In some situations there is actually no uncertainty, and a process can be fully formalized; in those cases we can automate the process, for example, in ISO 26262, there is a task that amounts to using a look-up table.

   Uncertainty analysis and interface contracts. Analysing uncertainty in WF^+ models depends on being able to make the uncertainty about processes explicit in the process models. We already show some examples of this in our WF^+ model with interface contracts. For example, the outcome of a revision process is uncertain until the process has actually been undertaken for a specific set of artefacts. Consequently, the post-conditions for such review processes state that the relevant output variable will be defined at the end of the process, but they do not state the value of the variable. More fine-granular capture of uncertainty is conceivable. For example, it is possible that a review process does not lead to a clear outcome. In other words, the output variable of such a process would remain undefined in such a case. Such uncertainty could be captured by using a richer logic for definedness—for example based on fuzzy set membership (Dubois & Prade 1980).

3. *Uncertainty in process models.* Epistemic uncertainty about process *models* appears to be of limited relevance; it may be that misunderstandings of what a WF^+ model actually represents could be considered epistemic uncertainty. In past work we have encountered several examples where after discussions with domain experts, we have discovered that the domain experts and users of WF^+ have different understandings of what the model meant.

   It is also worth briefly discussing aleatory uncertainty about process models. This occurs where it is inherently difficult to capture the diversity of processes *as performed* in a coherent process model *as specified* (Parnas & Clements 1986). Humans will inherently vary in their behaviours even for routine processes. This is even more the case the more complex a process becomes; the processes modelled in WF^+ tend to be fairly complex due to interactions and interleavings between the engineering tasks, the safety tasks and how organisations operate. As a result, it can be inherently difficult to provide process models that accurately represent every single variant of the process as performed. Some of this uncertainty is implicit in WF^+ models, which typically do not provide details of the steps in a process, for example. Other uncertainty of this type could potentially be captured by making explicit the range of possibilities—for example, using may-models (Famelis et al. 2013).

This analysis of uncertainty across artefacts and processes is important for the construction of safety cases. Safety cases, in essence, aim to argue that we have a sufficient level of certainty that the system will be safe – based on the development process, models, and analyses used in system development. This means, it is important that we can identify areas of uncertainty and their potential impact on safety analysis explicitly in our safety case. The above analysis, then, enables a more fine-grained analysis in the safety case: every combination of uncertainty

category and potential source of uncertainty implies the need of a sub-argument (a) acknowledging this form of uncertainty in the model and (b) discussing what has been done to mitigate the impact of the uncertainty on overall system safety. The above categorisation, thus, enables a more systematic approach to safety-case construction. For example, aleatory uncertainty in processes can be addressed in a safety case by (a) explicitly acknowledging that human review will have a subjective component (and potentially identifying specific ways in which this can manifest in a particular process) and (b) identifying what has been done to reduce the impact of subjective judgement (e.g., very carefully structured detailed guidance, or the use of multiple reviewers and a moderation process). This does not remove the uncertainty from the overall process, but it makes it explicit and allows its implications for system safety to be explicitly discussed and, where possible, mitigated.

5.2. Impact analysis

In our prior work with WF$^+$ (Annable et al. 2022), we briefly discussed how WF$^+$ models could be used to support change impact analysis (Briand et al. 2003), particularly with respect to assurance cases. Change impact analysis involves identifying property changes in one model, and calculating the resulting impact of those changes on any related models. For example, an assurance case might be automatically generated from a WF$^+$ model, and we might want to know the impact on the assurance case of refactoring a WF$^+$ data element into two distinct data elements that share a common parent. Because WF$^+$ provides explicit support for traceability between data, process and any generated assurance case, this enables a coarse form of impact analysis. In our previous project, impact analysis was calculated via queries on WF$^+$ models; these queries were implemented in the MMINT[3] tool and in Epsilon[4] using EOL. An example query might be to calculate which goals or strategies in an assurance case could conservatively be affected by changes to data encoded in a WF$^+$ model.

The impact analysis algorithms that have been implemented so far are, of course, based on queries applied to WF$^+$ models without interface contracts. As such, the impact analysis is at the level of coarse-grained changes to data (e.g., removing data elements, splitting data elements, renaming data elements) and process. By adding interface contracts, impact analysis can be much more fine-grained, and could potentially even take into account changes to the contracts, and their corresponding impact on an assurance case, into account. For example, a significant problem with impact analysis on large (WF$^+$) models is dealing with scope: a conservative impact analysis query may overestimate the extent to which the traced assurance case model is impacted by a change to a WF$^+$ data element. In the worst case, this could lead to a very significant sub-model of the assurance case being highlighted as potential impacted by a WF$^+$ model change. By taking into account interface contracts in impact analysis, it may be possible to further and more precisely restrict the scope of the impact analysis to a smaller subset of the traced assurance case. We are currently intending to explore these ideas in a new project that is building on the interface contract extensions we have presented in this paper.

[3] https://github.com/adisandro/MMINT
[4] https://www.eclipse.org/epsilon

6. Conclusion

WF$^+$ models provide a means for capturing the data and control associated with development of assurance. Their support for modelling activities, interrelationships, data and constraints is substantial, in terms of enabling the generation of assurance cases, but is nevertheless limited. This paper highlighted some of those limitations, particularly associated with capturing richer details associated with processes. To address this, the paper introduced a notion of interface contract for WF$^+$ which could be used to more precisely describe requirements and expectations associated with activities, and which could be used to support a notion of refinement of processes/workflows.

We illustrated some of the consequences of extending WF$^+$ with interface contracts, particularly for specifying and reasoning about uncertainty inherent in models. While our analysis of the types of uncertainty that could be revealed through extending WF$^+$ models with contracts, it was certainly systematic, and implies potential substantial benefit to this enrichment. Revealing uncertainty in WF$^+$ models has the added benefit of being able to obtain greater confidence in our models before they are used to, potentially, automatically generate assurance cases. In other words, this extension has the potential to give us greater confidence in the quality of our WF$^+$ models before they are used for downstream activities.

There are numerous directions for future work we are considering. Beyond further tool support for creating and editing WF$^+$ models (currently under development using Eclipse tools such as Sirius), we are exploring different mechanisms for adding hierarchies to WF$^+$ this, combined with interface contracts, would provide support for a discipline of refinement.

The analysis presented in this paper targets how interface contracts can be used to more extensively (and accurate) identify different kinds of uncertainty in WF$^+$ models. But there is another potential way in which this type of analysis can be used: to test existing classifications of uncertainty to determine their level of coverage and/or comprehensiveness. For example, it may be determined that a particularly uncommon type of uncertainty is not supported by an existing classification, by comparing concrete examples of uncertainty against what is supported by the classification. In this way, we could envision *testing* classifications, which in turn could give us greater confidence in their quality and accuracy.

The support for interface contracts in WF$^+$ models opens up new opportunities for other forms of analysis. We mentioned, in the last section, being able to support richer forms of impact analysis. As another example, current use of WF$^+$ models (Annable et al. 2022) has generally focused on capturing processes and workflows *as documented*, e.g., conforming to ISO26262. Given that, in engineering practice, the process as documented may differ from the process as actually implemented (Parnas & Clements 1986), it would be interesting to use WF$^+$ to model both processes and to rely on interface con-

tracts to help to automatically generate a difference model that expresses how the models – and hence, how the two versions of process – compare. This could in turn tell us something about the actual practice of compliance with standards.

Acknowledgments

We would like to thank the reviewers of this paper for their helpful comments and suggestions. We acknowledge the contributions of Zinovy Diskin, Alan Wassyng, Mark Lawford and our colleagues at the University of Toronto (specifically Marsha Chechik and Alessio Di Sandro).

References

Annable, N. (2020). *A model-based approach to formal assurance cases* (Unpublished master's thesis). McSCert, Dept. of Computing and Software, McMaster Univ.

Annable, N., Chiang, T., Lawford, M., Paige, R. F., & Wassyng, A. (2022). Generating assurance cases using workflow+ models. In *To appear in Proc. SAFECOMP 2022*. LNCS, Springer-Verlag.

Bernardi, S., Famelis, M., Jézéquel, J.-M., Mirandola, R., Palacin, D. P., Polack, F. A. C., & Trubiani, C. (2021). Living with uncertainty in model-based development. In R. Heinrich, C. Talcott, F. Duran, & S. Zschaler (Eds.), *Composing model-based analysis tools*. Springer.

BPMN 2.0 handbook second edition: Methods, concepts, case studies and standards in business process management notation. (2011). Future Strategies Inc.

Briand, L. C., Labiche, Y., & O'Sullivan, L. (2003). Impact analysis and change management of UML models. In *19th international conference on software maintenance (ICSM 2003), the architecture of existing systems, 22-26 september 2003, amsterdam, the netherlands* (pp. 256–265). IEEE Computer Society. Retrieved from https://doi.org/10.1109/ICSM.2003.1235428

Chiang, T. (2021). *Creating an editor for the implementation of workflow+: A framework for developing assurance cases* (Unpublished master's thesis). McSCert, Dept. of Computing and Software, McMaster Univ.

Dai, Y. S., Xie, M., Long, Q., & Ng, S. H. (2007). Uncertainty analysis in software reliability modeling by bayesian approach with maximum-entropy principle. *Transactions on Software Engineering*, *33*(11), 781–795.

Dubois, D., & Prade, H. (1980). *Fuzzy sets and systems: theory and application*. Academic Press.

Esfahani, N., & Malek, S. (2013). Uncertainty in self-adaptive software systems. In *Software engineering for self-adaptive systems II* (pp. 214–238). Springer.

Famelis, M., Salay, R., Sandro, A. D., & Chechik, M. (2013). Transformation of models containing uncertainty. In A. Moreira, B. Schätz, J. Gray, A. Vallecillo, & P. J. Clarke (Eds.), *Model-driven engineering languages and systems - 16th international conference, MODELS 2013, miami, fl, usa, september 29 - october 4, 2013. proceedings* (Vol. 8107, pp. 673–689). Springer. Retrieved from https://doi.org/10.1007/978-3-642-41533-3_41

ISO 26262. (2018). *Road vehicles – Functional safety*. ISO.

Jøsang, A. (2016). *Subjective logic: A formalism for reasoning under uncertainty*. Springer Cham. doi: 10.1007/978-3-319-42337-1

Jousselme, A.-L., Maupin, P., & Bossé, E. (2003). Uncertainty in a situation analysis perspective. In *6th international conference of information fusion* (pp. 1207 – 1214). doi: 10.1109/ICIF.2003.177375

Laguna, M., & Marklund, J. (2004). *Business process modeling, simulation and design*. Prentice-Hall.

Lebo, T., Sahoo, S., McGuinness, D., Belhajjame, K., Cheney, J., Corsar, D., ... Zhao, J. (2014). *PROV-O: The PROV ontology*. http://www.w3.org/TR/prov-o/. Retrieved from http://www.w3.org/TR/prov-o/

Munoz, P., Pérez-Vereda, A., Moreno, N., Troya, J., & Vallecillo, A. (2021). Incorporating trust into collaborative social computing applications. In *International Enterprise Distributed Object Computing Conference (EDOC)* (pp. 21–30). IEEE.

Oberkampf, W. L., DeLand, S. M., Rutherford, B. M., Diegert, K. V., & Alvin, K. F. (2002). Error and uncertainty in modeling and simulation. *Reliability engineering and system safety*, *75*(3), 333–357.

Padulo, M., & Guenov, M. D. (2012). A methodological perspective on computational engineering design under uncertainty. In *European congress on computational methods in applied sciences and engineering* (pp. 7509 – 7528). Retrieved from https://eccomas2012.conf.tuwien.ac.at/

Parnas, D. L., & Clements, P. C. (1986). A rational design process: How and why to fake it. *IEEE Transactions on Software Engineering*, *SE-12*(2), 251–257. doi: 10.1109/TSE.1986.6312940

Patelli, E., Panayirci, H. M., Broggi, M., Goller, B., Beaurepaire, P., Pradlwarter, H. J., & Schuëller, G. I. (2012). General purpose software for efficient uncertainty management of large finite element models. *Finite Elements in Analysis and Design*, *51*, 31–48.

Rolland, C., Prakash, N., & Benjamen, A. (1999). A multi-model view of process modelling. *Requir. Eng.*, *4*(4), 169–187. Retrieved from https://doi.org/10.1007/s007660050018 doi: 10.1007/s007660050018

Troya, J., Moreno, N., Bertoa, M., & Vallecillo, A. (2021). Uncertainty representation in software models: a survey. *Software and Systems Modelling*, *20*, 1183-–1213. Retrieved from https://doi.org/10.1007/s10270-020-00842-1

UML 2.5.1 specification. (2017). OMG. Retrieved from https://www.omg.org/spec/UML/2.5.1/About-UML/

Vallecillo, A. (2019). Belief uncertainty in software models. *Modelling languages*. Retrieved from https://modeling-languages.com/belief-uncertainty-models-uml/

Zhang, M., Selic, B., Ali, S., Yue, T., Okariz, O., & Norgren, R. (2016). Understanding uncertainty in cyber-physical systems: A conceptual model. *LNCS*, *9764*, 247-–264.

Zschaler, S., Kolovos, D. S., Drivalos, N., Paige, R. F., & Rashid, A. (2010). Domain-specific metamodelling languages for software language engineering. In M. van den Brand, D. Gasevic, & J. Gray (Eds.), *Proc. 2nd int'l conf. on software*

language engineering (sle'09) (Vol. 5969). Springer. doi: 10.1007/978-3-642-12107-4

About the authors

Richard F. Paige is Joseph Ip Distinguished Engineering Professor in the Department of Computing and Software at McMaster University, Hamilton, Canada. He is also Honorary Professor of Enterprise Systems at the University of York, United Kingdom. His research interests are in Model-Driven Engineering, safety-critical systems, medical devices and systems, and automotive systems. He is on the editorial board for Software and Systems Modeling and the JOT Journal.

Fiona A.C. Polack is Professor and Head of Department at the University of Hull, UK. Her research interests are in Model-Driven Engineering and the engineering of fit-for-purpose complex systems simulation.

Steffen Zschaler is a Reader in Software Engineering at King's College London and director of MDENet, the expert network on model-driven engineering. His research interests are in the foundations of MDE (in particular, using graph transformations and enabling modularity in modelling), search-based optimisation of models, and the use of MDE for the development of agent-based simulations. He is on the editorial board for Software and Systems Modeling.

Thomas Chiang is a PhD student at McMaster University, Canada, where he is studying Model-Driven Engineering, mobile computing and safety. For his MASc research he led the development of prototype tools for WF^+, building on its categorical foundations.

Nicholas Annable is a PhD student at McMaster University, Canada, where he is studying automotive software engineering, Model-Driven Engineering and safety. For his MASc research he developed many of the MDE foundations for WF^+ and applied it to automotive software engineering examples.