Advanced Engineering Informatics 30 (2016) 143–156

Contents lists available at ScienceDirect

Advanced Engineering Informatics

journal homepage: www.elsevier.com/locate/aei

Improving process conformance with Industry Foundation Processes (IFP)

Behrooz Golzarpoor [a,*], Carl T. Haas [a], Derek Rayside [b]

[a] Department of Civil and Environmental Engineering, University of Waterloo, Canada
[b] Department of Electrical and Computer Engineering, University of Waterloo, Canada

ARTICLE INFO

Article history:
Received 4 June 2015
Received in revised form 9 February 2016
Accepted 12 February 2016

Keywords:
Best practices
Workflow management systems
Process conformance
Process interoperability
Project performance

ABSTRACT

Conformance with standard corporate and institutional processes and industry best practices are sought because of regulatory requirements and evidence that best practices lead to improved project performance. Automated workflow engine enabled Industry Foundation Processes (IFP) are introduced in this paper that facilitate process conformance through structural transparency, foundation process inheritance, and automated conformance checking. While IFP processes can be customized to suit particular project or corporate conditions, they need to conform to a standard core structure and thus behavior. This has been achieved through defining specific workflow inheritance rules and developing an automated structural process conformance checking algorithm. The algorithm has been developed based on graph theory fundamentals using a first-order logic language, which compares two workflows and detects the conformance of a customized one with its associated IFP. The developed algorithm has been functionally tested and validated with different structural settings of workflows with a number of critical cases. Its functionality has been demonstrated in this paper with an example of the commonly used process of RFI (Request for Information). A new construct is thus contributed that can help improve process conformance to industry best practices particularly in the architectural, engineering, and construction industry, leading to improved project conformance.

1. Introduction

Traditionally, information systems have played a vital role in managing a business, enterprise, or project by supporting improved decision making. They have been widely used for creating, organizing, storing, retrieving, manipulating, and distributing information, and have had a positive impact on productivity and performance. Over the years, however, their applications and scope have been expanded from conventional data-aware information systems, such as database management systems, to process-aware information systems, such as business process management (BPM) and workflow management systems (WfMS) [1,2]. Conventional data-centric information systems are still an important backbone of modern information systems [1], but today's information systems rely on efficient and effective processes and best practices.

In the domain of the construction industry, several research studies [3–9] have confirmed that utilization of information systems and adoption of best practices drive performance and productivity improvements. The more recent findings of Kang et al. [4,10], however, revealed that improvements of automated work processes via information systems is in fact the main driver of improved project performance, and thus signified the importance of well-defined processes and best practices. Based on a statistical analysis of 133 construction projects from the Construction Industry Institute Benchmarking and Metrics database, they concluded that using information systems without enough attention to practices has a limited benefit for project performance, but the combined adoption of best practices and employment of information systems has a more significant impact on project performance. Their study challenged the common belief of strong direct correlation between employment of information systems and improved project performance, and suggested shifting focus to improvement of work processes to be more efficient or effective by adoption of best practices.

In accordance to this need, this paper introduces the concept of Industry Foundation Processes (IFP), workflow processes with minimal yet essential features, that are based on industry best practices. They are enacted via workflow management systems, and are customizable and evolvable to more complex processes

* Corresponding author. Tel.: +1 (519) 888 4567x31169.
E-mail addresses: bgolzarpoor@uwaterloo.ca (B. Golzarpoor), chaas@uwaterloo.ca (C.T. Haas), drayside@uwaterloo.ca (D. Rayside).

http://dx.doi.org/10.1016/j.aei.2016.02.005
1474-0346/© 2016 Elsevier Ltd. All rights reserved.

suitable for specific corporate and project conditions. The system of IFP workflow processes facilitates development of a standard core for construction industry common processes to improve process conformance and interoperability. The paper then focuses on mechanisms and the methodology for maintaining the conformance of customized workflow processes with their associated IFP, and automating this process.

Customization of IFPs is supported in a controlled manner by defining particular workflow inheritance rules that restrict or allow certain structural changes in the workflow. The workflow inheritance rules ensure that the core structure and the control-flow of an IFP is preserved within the customized workflow. An automated conformance checking system is developed using Java and a first-order logic language to streamline the workflow conformance checking process. This system reads workflow definitions from Microsoft Visual Studio Workflow Designer, analyzes them for conformance, and visualizes the result. A prototype example of an IFP process for industrial construction projects, the Request for Information (RFI), is used along with the conformant and non-conformant customized versions of this process to demonstrate the usefulness of the system.

Integration of IFPs into information systems facilitates more systematic and consistent adoption of best practices throughout a project lifecycle and from project to project. Developing and applying a system of Industry Foundation Processes (IFP) can add value to the use of information systems. The expected end result is capital project productivity and performance improvements. Fig. 1 outlines this rationale.

2. Background

Conventional data-aware information systems evolved around centralized database management systems. Today's process-aware information systems facilitate interaction and collaboration of stakeholders via distributed systems [2]. Examples include advanced project management collaboration tools, enterprise resource planning (ERP) systems [11–14], workflow engines [15–17], electronic document management systems [18,19], knowledge-based information systems [20,21], and more specifically electronic product and process management (EPPM) systems [22,23].

An Electronic Product and Process Management (EPPM) system is a workflow management system specifically designed for managing large-scale construction projects. A workflow engine at the heart of an EPPM system facilitates enactment of workflow processes; a document management system supports several types of files and enables sharing and modifying various types of documents; and a collaboration management system enables project delivery by collaboration among several stakeholders. In addition, the kernel of an EPPM system typically offers services, such as format management, version control, indexing, search, security, and publishing. EPPM systems store and manage various types of information regarding the lifecycle of a project from inception and planning to execution and startup. These systems not only facilitate enactment of processes via workflow engines, but they also facilitate interaction of process stakeholders and tracking and auditing of process steps. For example, change management, deliverables management, or interface management processes typically involve the interaction of several stakeholders, such as contractors, subcontractors, suppliers, consulting firms, and the owner(s). An EPPM system provides the infrastructure for defining, modifying, enacting, and auditing such processes.

Conformance of processes to industry best practices, corporate rules and regulations, or service level agreements in such systems is becoming increasingly important. Although conformance has different aspects and can be defined based on various considerations, conformance checking techniques typically focus on the control-flow of a process, and analyze the order of steps involved to determine the conformance of the process with the expected behavior [24]. There are two primary types of conformance checking: (1) forward conformance in which the restrictions are enforced in the process design stage to prevent designing a non-conformant process and (2) backward conformance in which the steps and flow of work in an implemented process is examined to discover non-conformant behavior [25].

Implementation of processes on EPPM systems requires customization of processes for each construction project, because of unique project characteristics, including size, delivery method, execution plan, and organizational structure. For example, a change request is distributed among the individuals and organizations that are particular to the project, with approval thresholds particular to that project. So, practice implementation through EPPM systems improves conformance through transparency and automation, but the required customization tends to work against best practice conformance. A potential solution is introduced based on the Industry Foundation Processes (IFP) construct presented here.

3. Industry Foundation Processes (IFP)

We define Industry Foundation Processes (IFP) as a simple standard definition of common workflows with particular properties and components that facilitate process conformance and interoperability. IFPs are defined as structured processes, so that the sequence of activities and their execution constraints are fully defined. They simplify the integration of best practices into workflow management systems and support their consistent implementation throughout project lifecycle and from project to project.

Industry Foundation Processes are process-based and workflow-driven. They focus on the flow of information or work, while abstracting from execution constraints, such as data dependencies and resource constraints. They are defined in their simplest form, containing all the essential steps, but with no extra or redundant activity. As such, they are general enough to be extendable to many situations, yet simple and streamlined. The idea of IFPs is inspired by the ideas of abstraction, inheritance, and modularity in object-oriented programming languages.

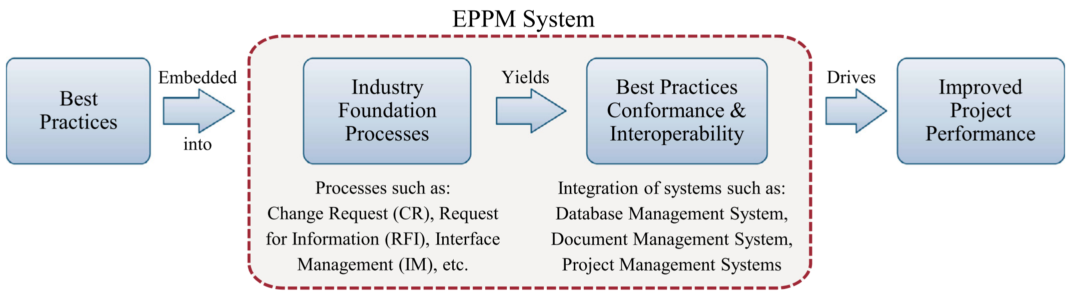

EPPM System
Best Practices
Embedded into
Industry Foundation Processes
Yields
Best Practices Conformance & Interoperability
Drives
Improved Project Performance
Processes such as: Change Request (CR), Request for Information (RFI), Interface Management (IM), etc.
Integration of systems such as: Database Management System, Document Management System, Project Management Systems

Fig. 1. IFP research rationale.

IFPs are abstracted to organizational-level processes, but include a minimum of operational-level details that are required for their enactment through workflow management systems. The workflow inheritance concept enables IFP workflow processes to be customized to more complex processes in a controlled manner to suit particular types and conditions of projects, while not losing their core structure. The system of Industry Foundation Processes may be defined for many common construction industry processes, such as change management, contract management, materials management, and deliverables management, as simple structured processes that incorporate the essence of best practices.

The use of IFP system has several practical advantages. It offers a standard core structure for implementation of common processes, and thus provides visibility to the core structure of complex processes. IFP system facilitates integration of best practices into workflow management systems, and improves process conformance and interoperability. It can be used to efficiently implement and manage systems of customized interoperable processes that conform to the best practices, and thus support improved project performance.

There are two primary approaches for deriving IFP processes: (1) define structured processes based on well-known best practices in the construction industry and (2) identify and extract the common core of different project/corporate implementations of a known process. The former approach involves exploring well-known construction industry best practices, developing high-level organizational processes that include the main steps for adopting those practices, transforming the organizational processes based on the roles and responsibilities of actors into structural processes implementable into workflow management systems, and defining IFPs based on the core structure of the structured processes.

The latter approach requires employing business process modeling tools and techniques to compare different implementations of a process that has been subject to the process improvement cycle over time, extracting a common core as a simple structured process, and defining an IFP based on that common core. For instance, Fig. 2 demonstrates a Request for Information (RFI) workflow that has been used in a recent mega-construction project in Canada. This workflow includes eight versions that have been revised and improved throughout the project lifecycle. RFI workflow is a method of requesting a design clarification, field construction clarification, or to provide supplemental instructions from either the project management team, or any company engaged in a construction project. The magnified portion shows some of core activities. The common core structure of the RFI process, based on an analysis of several implementations in different projects, is presented in Fig. 4.

4. Proposed IFP ontology

The system of Industry Foundation Processes offers a standard core for common construction industry processes in such a way as to improve process conformance to best practices and facilitate process interoperability. Based on synthesis of the literature, examination of functional and operational requirements for IFP system, and consultation with industry experts, this paper proposes an ontology for the IFP system which includes seven components (Fig. 3): (1) an applicable scope, (2) a core structure and functionality, (3) defined abstraction level to essential details, (4) associated data structures, (5) workflow inheritance property, (6) conformance to best practice, and (7) interoperability with other workflow processes. The last two — conformance and interoperability — are the end-result of using the IFP system and are dependant on the other components, but they also affect the structure and behavior of the process, and thus should be considered while developing an IFP process.

The ontology components will be discussed within the following sections, except the interoperability which will be discussed in more detail in a separate publication.

4.1. Scope

The domain of the IFP system, similar to the domain of the Industry Foundation Classes (IFC), is defined as the Architecture, Engineering, Construction, and Facilities Management (AEC/FM) industry; however, the concept of the IFP system and the development methodology can be adopted by any industry. In addition, each individual IFP workflow process (*i.e.*, contract management, change management, or material management) is defined for a particular type of project, and thus has a more specific scope. The scope of an individual IFP process is defined according to project type, size, delivery method, etc. For example, a change management IFP process developed for large-scale industrial oil and gas projects is different from a change management IFP process defined for smaller-scale residential or commercial projects. Each of these change management IFP processes then can be customized for specific projects.

4.2. Core structure

Any process has a core structure that includes essential activities and their relationships. Selecting a complex process, and repeatedly substituting its activities and relationships with more abstract ones, ends up with a set of activities and relationships that are minimal, but sufficient for representing the purpose of that process [26]. Additional activities and relationships are typically added to the core structure to customize the process for specific purposes or conditions, but if any of the essential activities and relationships removed, the meaning of the process might not be preserved. Extracting the minimal yet essential elements of a complex process, developed based on the industry best practices and improved over time through the process improvement cycle, results in the core structure required for defining an IFP process. For example, as outlined previously, extracting the core structure of implementation-level RFI processes, such as the process shown in Fig. 2, results in the minimal yet essential activities and relationships presented as a simple structured process in Fig. 4.

As such, the core structure of an RFI process includes the following steps. A project team member initiates a request (1). A coordinator verifies the request for accuracy and completeness (2), and assigns/confirms participants. If any clarification is necessary (3), the request is being sent to the initiator for clarification (4); if not, it is being sent to one or more participants (5), typically a lead engineer or a construction manager, for composing a response. The consolidator is then responsible for consolidating responses (6), approving and issuing the response to initiator (7); and finally all process stakeholders are informed and the workflow is closed (8).

4.3. Abstraction level

The abstraction level of a workflow process is important because it determines the amount of detail that the process is represented with. A process can be characterized in a high-level abstraction level that simply explains the process steps, or it can be defined as a structured process in which the sequence of activities and their execution constraints are completely defined. Furthermore, a process can be presented with operational details that include activities and their relationships, or it can be defined with implementation details that contain information on execution and technical details required for enactment of the process in a computerized system. Based on an examination of industry

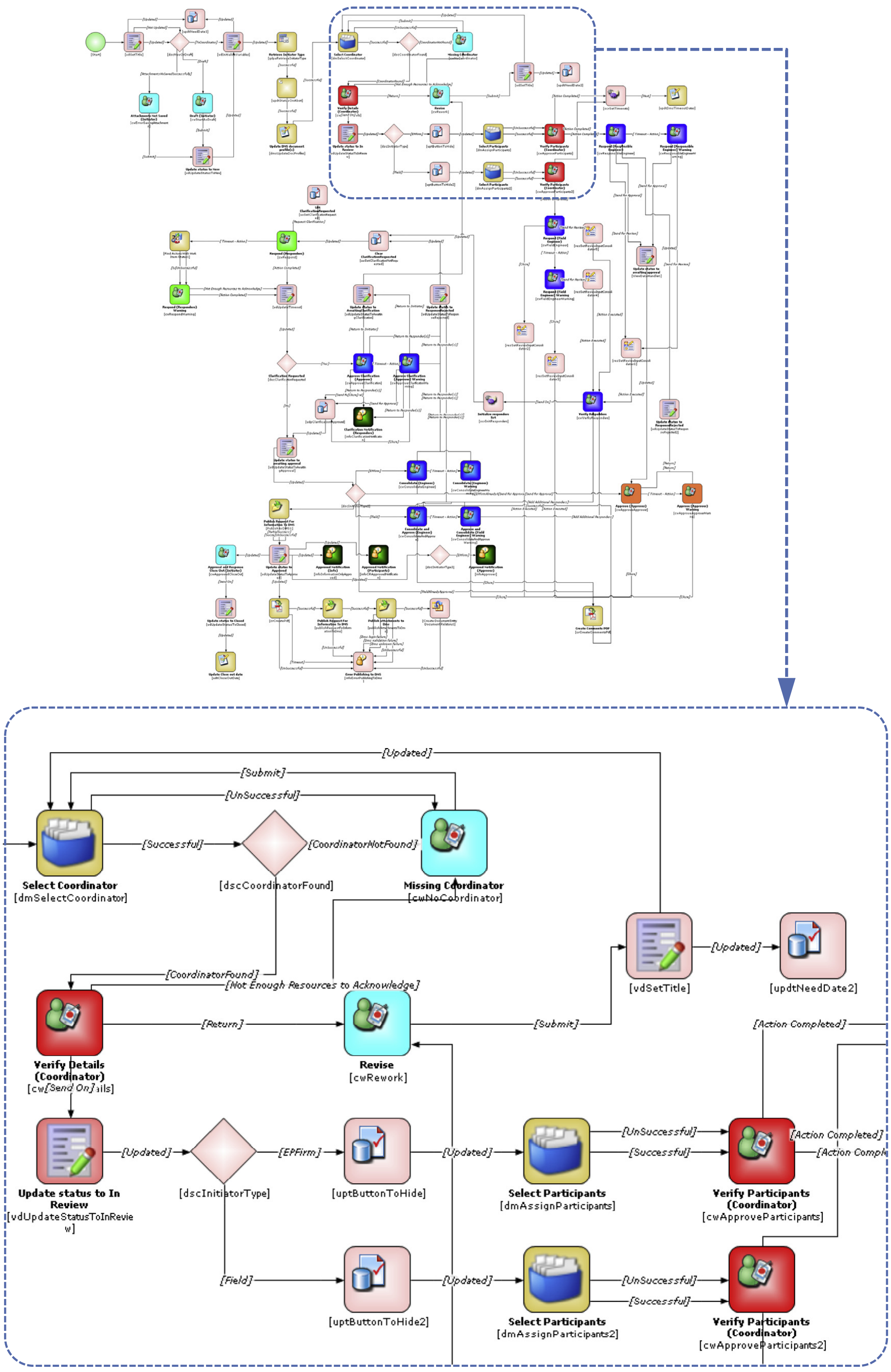


Fig. 2. An implemented version of the RFI process in Skelta software format as deployed by research partner.

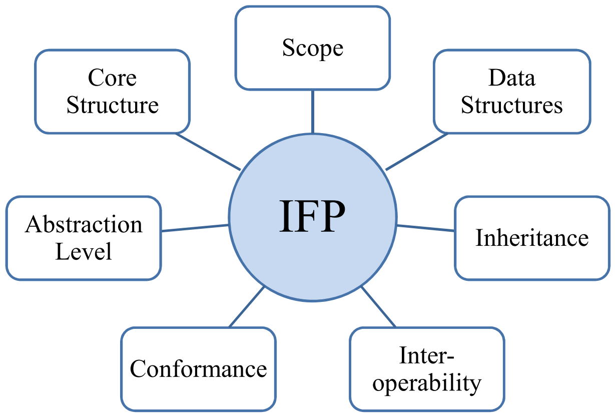


Fig. 3. Proposed IFP ontology.

Table 1
Workflow abstraction levels.

Abstraction level	Description
Meta-level workflow	A conceptual description of the process flow Includes organizational-level details
Foundation-level workflow	A high-level structured definition with particular properties Includes operational-level details
Workflow template	A customized workflow with the most common components Includes operational-level details
Workflow implementation	An implemented workflow for a specific organization or project Includes implementation-level details
Workfow instance	An executed instance of an implemented workflow Includes implementation-level details

practices, an analysis of the literature, and the required level of details for the IFP system, this paper classifies workflow processes into the following five abstraction levels (Table 1), and proposes the foundation-level as the appropriate level of abstraction for IFP processes: (1) meta-workflows, (2) foundation-level workflows, (3) workflow templates, (4) workflow implementations, and (5) workflow instances.

A meta-workflow is a conceptual definition of a workflow, either textual or in a flow-charting format. It is not a structured definition of a workflow and its main purpose is to describe the workflow behavior. A foundation-level workflow, associated with the concept of Industry Foundation Processes, however, is a structured definition of a process, with some operational and implementation level details that are required for its proper functioning. It is the highest abstraction level implementable in a workflow engine enabled environment, such as Skelta or Microsoft Workflow Foundation, which are the environments used in this research. A workflow template is a customized workflow, based on an IFP, that contains the most common activities and relationships for a particular type of project. It can be used as the starting point for deriving more detailed implementation-level workflows suitable for a specific project. Workflow implementations typically include all the required human-oriented tasks, as well as automated tasks, such as writing to databases and sending notifications to participants, as required.

An executed version of an implementation-level workflow is called a workflow instance. For any implementation level workflow, several workflow instances are typically created throughout the lifecycle of the project. Some workflow instances might have a relatively short lifetime, and some might be active for a longer period of time, before completing their execution and closing out. Each workflow instance typically stores all the data associated with its execution steps. For example, the execution details of an activity called "Verify Details", which is one step within the RFI workflow, include details such as, instance identification code, accessed time and date, completed time and date, name of responsible and responding party, current status of workflow, and more. All workflow instance execution data are stored in databases for retrieval and analysis, for auditing purposes, or to improve the definition of the workflow.

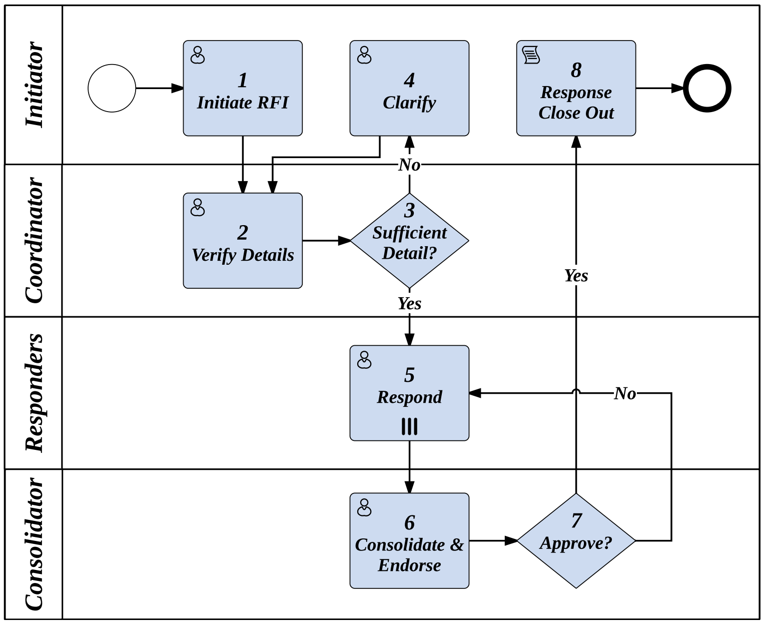


Fig. 4. The core structure of an IFP for the RFI process.

As an example, an IFP process for deliverables management with the domain of AEC/FM and the scope of industrial projects, can be customized to a deliverables management workflow template suited for oil and gas projects, and then customized and implemented for a specific project, with several instances of the workflow running simultaneously on a workflow management system.

4.4. Data structures

Processes rely on particular data structures for their proper functioning throughout the execution steps. A process stores, manipulates, and passes information with the flow of work from one step to another. For example, the execution of a request for information (RFI) process requires data fields, such as RFI ID, Contract ID, Title, Description, Request Date, and Response Date. Some data structures are being manipulated within the subsequent execution steps, such as "Response Note", and some of them even determine the flow of work while executing the process. For example, the flow of work might be redirected to a different person depending on the time or cost impact of the request. An IFP is defined with a minimal set of data structures that are required for its proper implementation. Table 2 presents a minimal data set that is associated with an RFI process.

Some of the data and metadata fileds are automatically assigned by the workflow management system, *i.e.* Process ID, Response Date, and Approve Date, and some of them are entered by process participants in each step of the process. Additional fields can be added when required, but the minimal set that is defined within an IFP is kept while customizing a process.

4.5. Inheritance

In computer science, inheritance is a key programming concept. Inheritance enables reuse of code by keeping certain properties of an object called a super-class, while transforming it into a new object called a sub-class. Sub-classes typically include extra or more detailed features. The inheritance concept can be applied to the IFP system whereby the core structure and particular properties of an IFP workflow is inherited, and additional activities or properties are added to form a customized version of the workflow. The idea of using the inheritance concept for workflow processes is not new. Van der Aalst explored the concept of workflow inheritance [27–29], and developed four types of workflow structural inheritance: protocol, projection, protocol/projection, and lifecycle inheritance. A detailed description of these workflow inheritance notions is beyond the scope of this paper, and the reader is referred to the cited references for more information.

This paper offeres three categories of inheritance for workflow processes to facilitate conformance with regulatory requirements or institutional practices: (1) structural, (2) organizational, and (3) temporal, and defines sets of workflow inheritance rules for structural and organizational inheritance to allow or restrict certain workflow transformations. These inheritance rules control how more detailed implementation-level processes are derived from an IFP, while maintaining conformance to the IFP. Structural inheritance rules restrict the flow of work or information in subclasses of a workflow to the sequence and set of core activities defined in a superclass IFP. This ensures that the core structure of an IFP process does not change when it is customized to suit specific projects.

Organizational inheritance rules ensure that the level and sequence of authorization defined in an organization or project is met with the execution of the workflow process. For example, if someone is not available who would be the next responsible person to whom the work or information be directed, or who could be assigned as a delegate for somebody who is not available for a period of time. For this purpose, a responsibility assignment matrix, *i.e.* a RACI chart is used to define the participation of various process stakeholders with their defined roles, responsibilities, and deliverables in completing each step of the process. RACI is an acronym that stands for Responsible, Accountable, Consulted, and Informed. A sample of a RACI chart is presented in Table 3.

Temporal inheritance rules define allowable durations for each activity according to regulatory or contractual obligations or industry best practices. For example, how much time is allowed for an approval activity to be finalized according to regulatory, institutional, or contractual obligations. Table 4 presents a set of structural inheritance rules to preserve the presence and the sequence of core activities in a customized workflow process. In addition, it offeres a sample of organizational inheritance rules. Organizational and temporal rules are defined as properties associated with the core structure of an IFP process, and thus the structural inheritance rules are the most important rules for conformance checking. In this paper we focus on the structural inheritance rules.

Fig. 5 graphically presents accepted and prohibited transformations for a simple specification workflow $A \rightarrow B \rightarrow C \rightarrow D$ in which the flow of work is only possible through A then B then C and then D. As demonstrated in Fig. 5(a), it is accepted for the super-class specification workflow of $A \rightarrow B \rightarrow C \rightarrow D$ to be transformed into sub-class workflows presented as W1 through W5. In all of these transformations none of the core activities can be skipped or their sequence be altered. W2 represents dividing an activity into two, in which part of the enactment of task B is performed in task B1 by one person, and the rest is performed in B2 by someone else.

Fig. 5(b) presents a set of transformations for the specification workflow $A \rightarrow B \rightarrow C \rightarrow D$ that are prohibited according to the defined workflow inheritance rules. Sequence of activities should not be changed (W6). Parallel paths are not allowed (W7, W8) by which the execution of some core activities might be circumvented. While new blocks of activities might be added between two adjacent existing activities, they should not be connected to any successor activities (W9, W10). For instance, W3 is an accepted transformation, but W10 is not. The inheritance rules ensure that all the core activities are present, and the sequence of their execution is not altered. Workflow inheritance is a key feature of Industry Foundation Processes. It enables reusability and customization of IFPs for different project circumstances, and is a basis for IFP conformance and interoperability.

Table 2
Minimal set of data structure fields for an RFI process.

RFI ID	Title	Request reason
Contract ID	Description	Need date
Project ID	Unit	Respondor
Request type	Area	Response note
Requested by	Discipline	Response date
Request date	System	Coordinator
Cost impact	Status	Approve date
Schedule impact	Priority	Final response

Table 3
Sample of a RACI chart.

Activities	Role 1	Role 2	Role 3	Role 4	Role 5	Start	Finish
Activity 1	I	R	A	A	I	10-March	18-July
Activity 2	R	I	A	A	I	11-September	15-December
Activity 3	I	I	R	I	C	14-September	16-November
Activity 4	A	R	I	I	A	12-October	03-December

Table 4
Sample IFP inheritance rules to ensure conformance with regulatory requirements or institutional practices.

Category	Inheritance rules
Core activities	Core activities should not be removed, *e.g. request, verify details, respond, and approve in an RFI process*
	The sequence of core activities should not be modified (W6)
	A connection from an activity to any of its predecessor activities might be added (W1)
	One core activity may be distributed into two or more activities (W2), *e.g. a double-stage approval*
Additional activities	Additional activities might be added between core activities (W3)
	Additional activities should not create a parallel path in the workflow (W7, W8)
	But, additional activities might bring the flow to a predecessor activity (W4)
	An additional activity can be in relationship to one activity (W5)
Roles & responsibilities	Extra roles might be added
	A lower-ranked role cannot approve the work of a higher-ranked role
	Responsibilities of a role might be delegated to another role
	Different roles might have the same responsibility

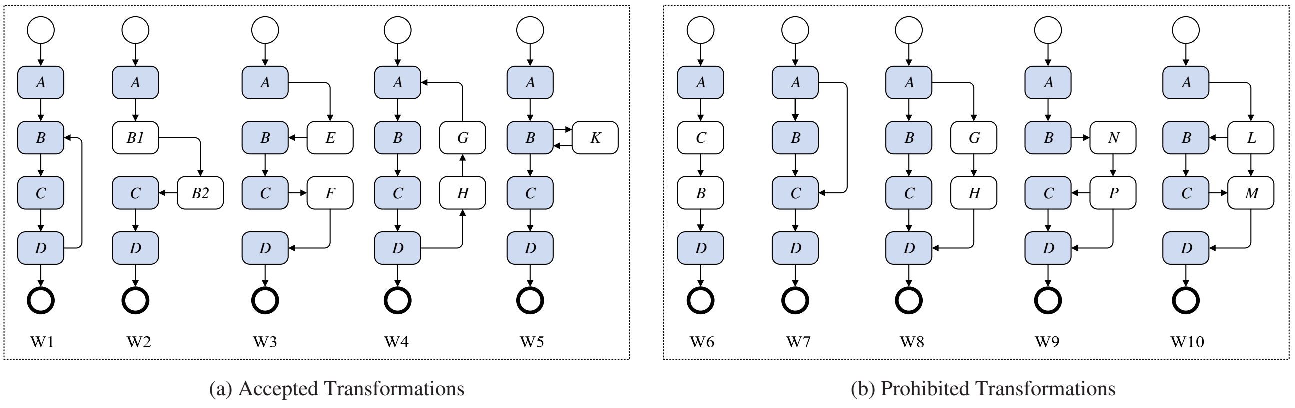


Fig. 5. Examples of accepted and prohibited transformations.

4.6. Conformance

Conformance of customized complex processes to their associated IFP process facilitates transparency, and streamlines process improvement and reengineering. IFP inheritance as a key property of the IFP system provides a method for systematic evolution of IFP processes into more complex customized implementations for a specific project, while maintaining conformance to requirements. Enforcing the inheritance rules at the workflow design stage ensures that sub-classes of a particular workflow are in conformance with its associated IFP. This is called forward conformance checking. On the other hand, a workflow process can be designed with no structural restrictions at the design stage. In this case the customized version of a workflow can then be compared with its associated IFP according to the inheritance rules, to discover whether it is in conformance or not. This is called backward conformance checking.

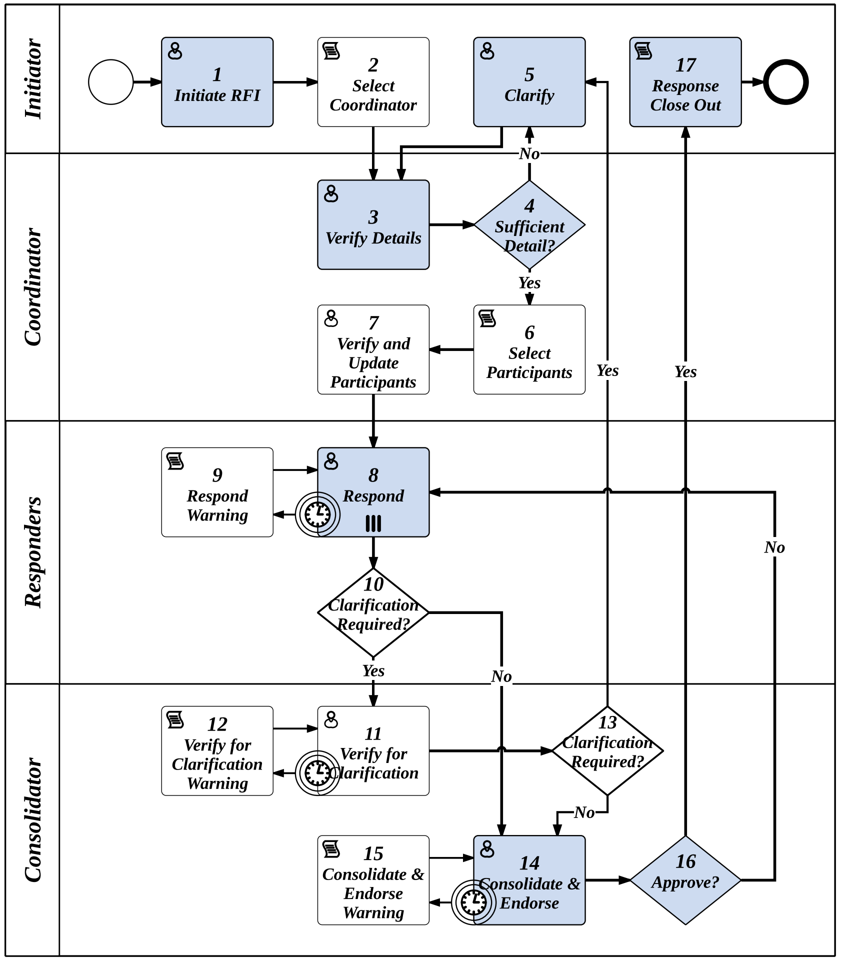
For instance, Fig. 6(a) demonstrates a customized version of the RFI process which is in conformance with the RFI-IFP process presented in Fig. 4. However, based on the defined workflow inheritance rules the workflow demonstrated in Fig. 6(b) is not in conformance with the RFI-IFP process, because of the direct connection between activity 6 and activity 18 which creates a parallel path. In Fig. 6, all the core activities that are associated with the set of activities available in the IFP process are outlined in gray. The additional activities are outlined in white. The backward conformance checking is not an easy task for complex implemented versions of processes, and thus cannot effectively be guaranteed. In this paper, we present a practical solution for automated backward conformance checking of workflow processes using a first-order logic language.

5. Validation (functional demonstration)

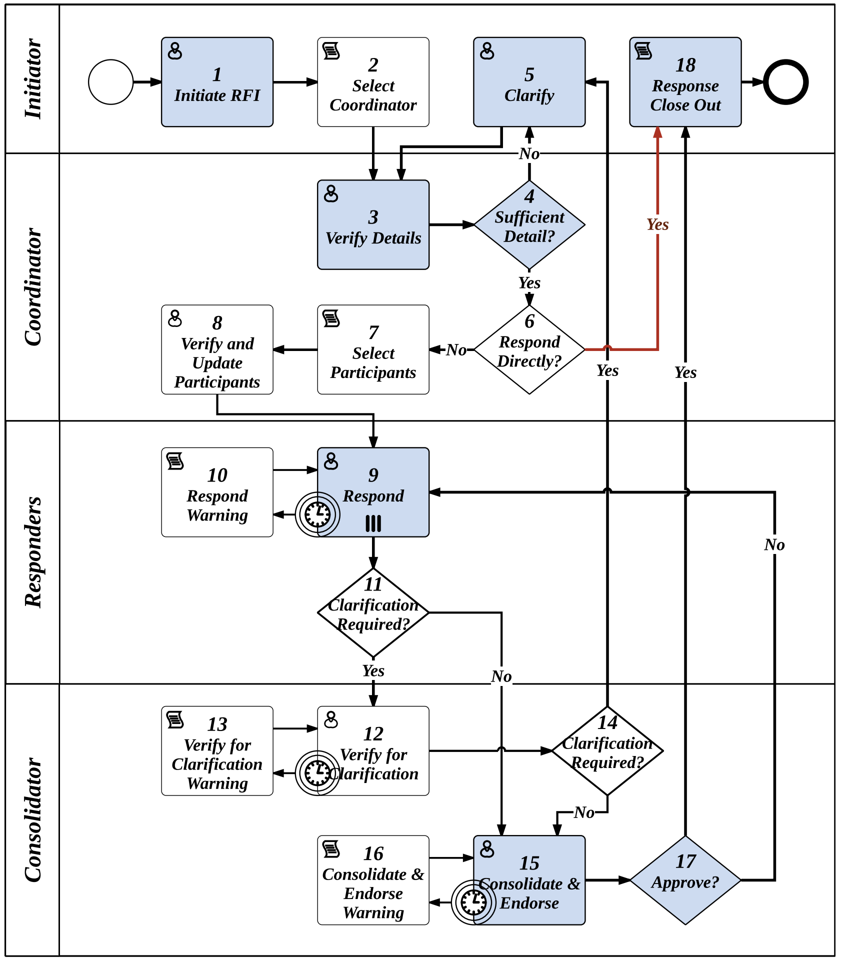
To illustrate the deployment process for the IFP system and to validate its functionality and benefits, this paper presents: (1) an implementation of the RFI-IFP process into a workflow management system and its evolution and customization to workflow processes demonstrated in Fig. 6, using both workflow inheritance rules and object-oriented programming inheritance constructs and (2) an application of the principles of the workflow conformance checking according to the defined workflow inheritance rules, using a first-order logic language and its associated analyzer tool. Validation experiments are conducted using over twenty test cases.

6. IFP deployment using Workflow Foundation (WF) technology

The IFP system can be deployed via any of several available open-source or commercial workflow management systems, such as *Activiti, IBM BPM, SAP Business Workflow, Skelta BPM*, and several others. The *Workflow Reference Model* [30] defines the general specifications of workflow management systems. In this paper we have used Microsoft *Windows Workflow Foundation (WF)* technology for deployment of the IFP system. WF technology is a component of the .NET Framework in Microsoft Visual Studio. WF 4.5 offers a declarative programming environment in which the code is separated into programming chunks defined as activities. Each activity is defined as a class, and the flow of code is modeled as a workflow through the objects of these classes. This model driven development is especially useful for managing complex applications and large codes, to avoid losing the outline of the program through the code details [31,32].



(a) An RFI Process in Conformance with the IFP



(b) An RFI Process Not in Conformance with the IFP

Fig. 6. Conformant and non-conformant versions of the RFI process.

The model is then executed by a runtime engine. The runtime engine or more specifically the Common Language Runtime (CLR), not only manages the memory but also provides control for asynchronous (execution of code in a separate thread of CPU) and parallel execution in a distributed system [33].

WF in the .NET Framework 4.5 offers three control flow structures: sequence, flowchart, and state-machine. The sequence workflow model defines the flow of code as a sequence of activities. The flowchart contains flow control elements and is typically used to implement non-sequential workflows. In the flowchart model, the flow of execution of activities is based on the values of variables. State-machine provides an alternative approach to model the flow of events that cannot usually be predicted in advance. This approach relies on states and transitions between states, and is suitable for modeling workflows that involve human interactions [31,34].

C# programming language along with Microsoft Windows Workflow Foundation (WF) has been used to implement three versions of the RFI workflow: an RFI-IFP workflow that is shown in Fig. 4, and two more detailed RFI workflows presented in Fig. 6, one in conformance with the IFP workflow and one that is non-conformant. For the implementation of workflows, the state-machine model is used. A graphical representation of the model for the RFI-IFP prototype is presented in Fig. 7.

To derive more detailed workflows from an IFP, both traditional programming inheritance which is part of object-oriented programming languages as well as the workflow inheritance which is based on the defined inheritance rules are required. The programming inheritance is used whenever a new activity can be defined as part of an existing activity. For example, in Fig. 6(a) activity 9 is an automated warning to activity 8, and thus can be implemented inside activity 8 using programming inheritance available in the programming language – in this case C#. However, in many cases the new activity cannot be merged into its predecessor or successor, *i.e.* when the new activity is performed by a different role, or when the nature of work performed by the new activity is different. For example, activity 7 in Fig. 6(a) cannot be merged into activity 8, because they are being performed by different roles (people). In such cases the programming language inheritance is not sufficient – the workflow inheritance is then used.

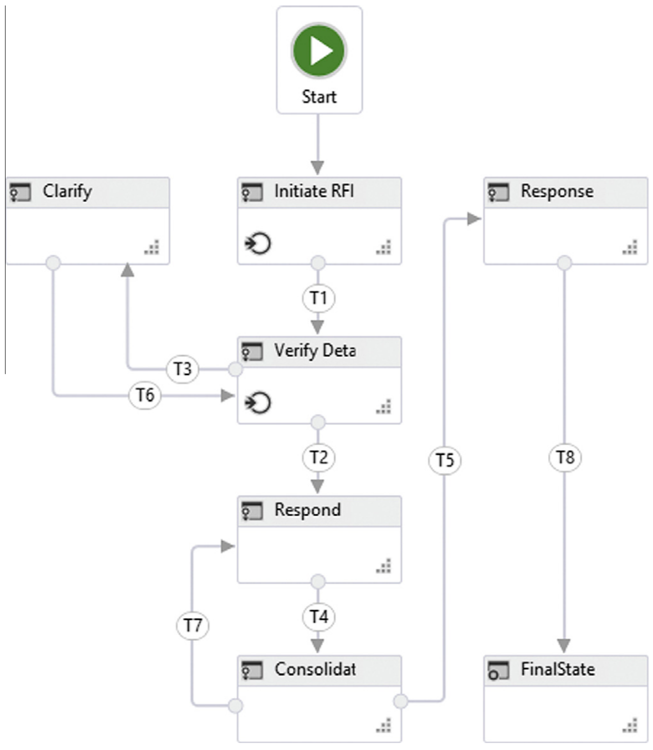


Fig. 7. Implementation of the RFI-IFP workflow as a state machine model.

Construction industry workflow processes, such as the RFI process, are typically enacted over distributed systems. The flow of work or information is being sent to different stakeholders who are able to login and perform one or more steps of the workflow. Microsoft Windows Workflow Foundation fully supports parallel and distributed computing and is a suitable platform for developing distributed systems. In WF 4.5 the process logic is defined as a workflow which is executed by the runtime engine. The selected project type in Visual Studio determines the type of application (distributed or centralized) and the required user interface. Console applications print on the default console, but windows-client applications and web-client applications require their own user interfaces.

Web-client applications which are used for distributed systems are the most representative implementation for the RFI process. However, to scale down the complexity for functional demonstration and validation, a windows-client desktop application has been developed using Microsoft Workflow Foundation technology and C# programming language. Since we have used classes in the desktop implementation that are typically used in distributed systems for long-lasting processes, the implementation would not be considerably different on a distributed setting, and thus the validation is realistically valid even for a distribute system.

While the application is running in the system, several instances of the RFI workflow process can be initiated and be enacted simultaneously. Different users can login and complete their associated tasks. When an instance is waiting for response, it is unloaded from memory to a database, and as soon as a response from a process stakeholder is received, it is loaded to memory and continues its execution steps. The process model is saved in a XAML file – a type of XML file developed by Microsoft. This XAML file is used as an input to the automated workflow conformance checking tool, as described next.

7. Automated workflow conformance checking

Workflow inheritance/conformance is formally defined in terms of the *graph dominators* concept from compilers [35–37]. The dominators concept applies to directed graphs with distinguished start and end nodes. The graph must be connected: that is, every node must be reachable from the start, and the end must be reachable from every node. These conditions are true of well-formed computer control flow graphs and also of workflow graphs.

A node *d dominates* node *n* if every path from the start node to *n* must go through *d* [35–37]. Similarly, a node *p post-dominates* node *n* if every path from *n* to the end node must go through *p* [35–37]. The *immediate dominator* is the dominator closest to the node. Similarly, the *immediate post-dominator* is the post-dominator closest to the node. Graph dominators can be computed in quadratic time [36].

A customized workflow is said to conform to a specification workflow if the following three conditions are met:

1. The customized workflow contains all of the steps (nodes) in the specification workflow.
2. For every step *X* that exists in both the specification and customized workflows, *X*'s immediate dominator in the specification workflow is one of its dominators in the customized workflow.
3. For every step *Y* that exists in both the specification and customized workflows, *Y*'s immediate post-dominator in the specification workflow is one of its post-dominators in the customized workflow.

These conditions formalize the intuitions that steps in the specification workflow cannot be skipped and that new steps may be added. Using these conditions, an edge e from source node s to target node t is classified as a *skip* edge if either the source node s fails to meet condition 2 above, or the target node t fails to meet condition 3.

An algorithm to assess workflow conformance is as follows:

1. Confirm that both the specification workflow and the customized workflow are well-formed. If not, report malformed workflow and terminate.
2. Confirm that the customized workflow contains all of the steps in the specification workflow (condition 1). If not, report non-conformance due to step deletion and terminate.
3. Compute dominators and post-dominators for every node, in both the specification workflow and the customized workflow.
4. For every node that exists in both specification and customized workflows, check that the immediate dominator in the specification workflow is still a dominator in the customized workflow (condition 2 above). If not, report edges that terminate at such nodes as skip edges.
5. For every node that exists in both specification and customized workflows, check that the immediate post-dominator in the specification workflow is still a post-dominator in the customized workflow (condition 3 above). If not, report edges that originate at such nodes as skip edges.
6. If no skip edges, then report conformance.
7. Terminate.

This algorithm has been implemented in the Alloy logic language [38], so that specific pairs of workflows can be automatically checked for conformance using the associated Alloy Analyzer tool. An alternative implementation could be written in a conventional imperative programming language (*e.g.*, Java, C, etc.) using one of the well known algorithms for graph dominators (*e.g.*, [36,37]).

7.1. The Alloy language

Alloy has three advantages over a conventional imperative language for this task. First, the Alloy language is designed for working with rich graph-like structures, whereas conventional imperative programming languages are not (SETL is a notable exception [39]). Second, the Alloy Analyzer includes a visualizer for inspecting the inputs, outputs, and state of the program. Finally, in addition to running the program with specific inputs, the Alloy Analyzer can also automatically generate test inputs for sub-procedures or the program as a whole.

The Alloy language is a first-order logic with sets, relations, and transitive closure. It is typically used for writing specifications of rich graph-like data structures, which are structurally similar to workflows. The Alloy Analyzer translates the Alloy first-order logic to propositional logic (*i.e.*, Boolean formulas) by providing finite bounds for the quantifiers. If the finite bounds used for translation are insufficient, then the resulting Boolean formula is an approximation of the original first-order formula. For example, if the original formula quantifies over an infinite set such as the integers then the bounds will be insufficient. Since workflows are always finite structures, and from a computational standpoint not particularly large, the bounds for the translation can always be adequate for workflows.

The computational complexity of computing dominator trees is merely quadratic [36], which is well within the expressiveness of Boolean formulas (NP-complete [40]). So the Boolean formula produced by the Alloy Analyzer is a faithful representation of the problem of computing the conformance of two workflows. Modern Boolean Satisfiability solvers routinely solve formulas with tens of thousands of variables and hundreds of thousands of clauses. The Boolean formulas produced for workflow conformance checking typically have several thousand variables and several thousand clauses, and solve in a few tenths of a second using MiniSAT [41] on an old laptop (AMD A4-3300 M processor running at 1.9 GHz; manufactured in 2011). Workflow conformance checking is well within the capabilities of modern SAT solvers.

In software engineering, Alloy is used for analyzing software designs, including analyzing imperative programs for conformance with their logical specifications [42]. The workflow-specification conformance problem is similar to, but importantly different from the program-specification problem: most importantly, workflow conformance checking is only concerned with the arrangement of the steps, and not with the outputs of the workflow. Program-specification checking is concerned with the outputs computed by the program. Workflows involve highly trained people exercising professional judgments in complex real-world situations, rather than computers merely following instructions. The workflow-specification conformance problem is more similar to the subgraph isomorphism problem [43]: are the steps of the specification workflow embedded in the customized workflow in a way that preserves their ordering? The subgraph isomorphism problem is simplified here by fixing the node correspondences based on the node labels. Order preservation is relaxed from the subgraph isomorphism problem by permitting the insertion of nodes and the insertion and removal of edges. Permissible order-preserving modifications are formalized in terms of dominators and post-dominators.

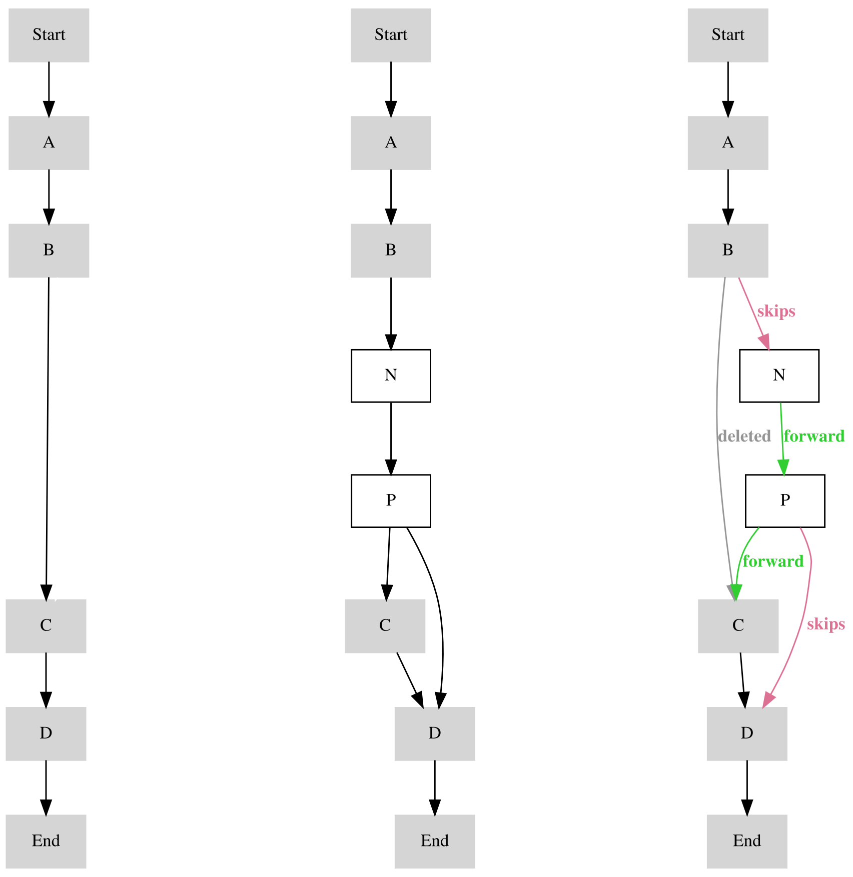
7.2. An example

Fig. 8 shows the Alloy visualization of the conformance check of example workflow W9 from Fig. 5. Fig. 8a shows the specification workflow (ABCD). Fig. 8b shows the customized workflow (W9). Fig. 8c shows the conformance analysis. The gray nodes are those that exist in both the specification and customized workflows. The white nodes are new nodes in the customized workflow. Black edges are those that exist in both workflows. Gray edges exist in the specification workflow, but have been deleted in the customized workflow ($B \rightarrow C$). Green edges are new legal forwards edges in the customized workflow ($N \rightarrow P, P \rightarrow C$).

As discussed above, workflow W9 does not conform to the specification workflow. This is illustrated in the analysis by the red skip edges in the analysis: $B \rightarrow N$ and $P \rightarrow D$ (Fig. 8c). The dominator subset analysis reports that node B has a problem because in the specification workflow B's post-dominators include C and D; whereas in the customized workflow B's post-dominators include D, N, and P: it is fine to add N and P, but not to remove C. Similarly, the dominator subset analysis reports that node D has a problem because in the specification workflow D's dominators include A, B, and C; whereas in the customized workflow D's dominators include A, B, N, and P: it is fine to add N and P, but not to remove C. As a consequence of this dominator analysis, the edges $B \rightarrow N$ and $P \rightarrow D$ are reported as skip edges.

7.3. Code listing

Fig. 9 lists an excerpt of the Alloy code for workflow conformance checking that shows the classification of skip edges based on the dominator analyses. An expression such as `n.idom1` evaluates to the immediate dominator of node n in the specification workflow. An expression such as `n.^idom2` evaluates to the set of all dominators of n in the customized workflow. Here `idom1` and `idom2` are functional binary relations that map nodes to their immediate dominator in the specification or customized workflow,



(a) Specification Workflow (Sec. 4.5) *(b)* Customized workflow (Figure 5b:W9) *(c)* Overlay Analysis

Fig. 8. Visualization of conformance check for workflow W9 (Fig. 5).

```
fun skips[] : Step →Step { { s,t : Step |
 −− source →target is an edge in the customized workflow and
 s→t in v and (
   −− target's original immediate dominator is not in its new dominators
   t.idom1 not in (s +t.ˆidom2)
   or
   −− or source's original immediate post−dominator is not in its new post−dominators
   s.ipostdom1 not in (t +s.ˆipostdom2)
) } }
```

Fig. 9. Alloy specification (excerpt) of workflow conformance for steps 4 and 5 of the algorithm.

respectively. The tilde (∧) operator computes the transitive closure of a binary relation: *i.e.*, finds the entire set of dominators. This code is more concise in Alloy than it would be in a conventional imperative programming language.

7.4. Case study

Fig. 10 shows the overlay analysis and visualization of the customized RFI workflow from Fig. 6b with respect to the IFP specification workflow in Fig. 4. Fig. 10 was generated automatically, from XAML files representing the RFI and IFP workflows of Figs. 6b and 4, respectively. The XAML files are automatically translated to Alloy and then checked for conformance.

The visual conventions in Fig. 10 are the same as in Fig. 8: gray nodes are those in the specification workflow; white nodes are those added in the customization; black edges exist in both workflows; gray edges are those that have been removed in the customization; green edges are new forward edges; blue edges are new back edges; red edges are skips. One of the purposes of this customization was to add the path to RespondDirectly, bypassing the ConsolidateAndEndorse and Approval steps. Of course bypassing steps is not permitted, so this customization is deemed to be non-conformant with the specification (Fig. 4). The Alloy implementation correctly identifies the edge from RespondDirectly to ResponseCloseOut as the skip edge.

In summary, the automated workflow conformance checking tool is comprised of three components: (1) Workflow Designer, (2) Workflow Analyzer, and (3) Visualizer, which all work well together to streamline the process of workflow conformance checking. Specification and customized workflows are being designed in Visual Studio Workflow Designer and are stored as XAML files. A Java application converts XAML files (state-machine or flowchart)

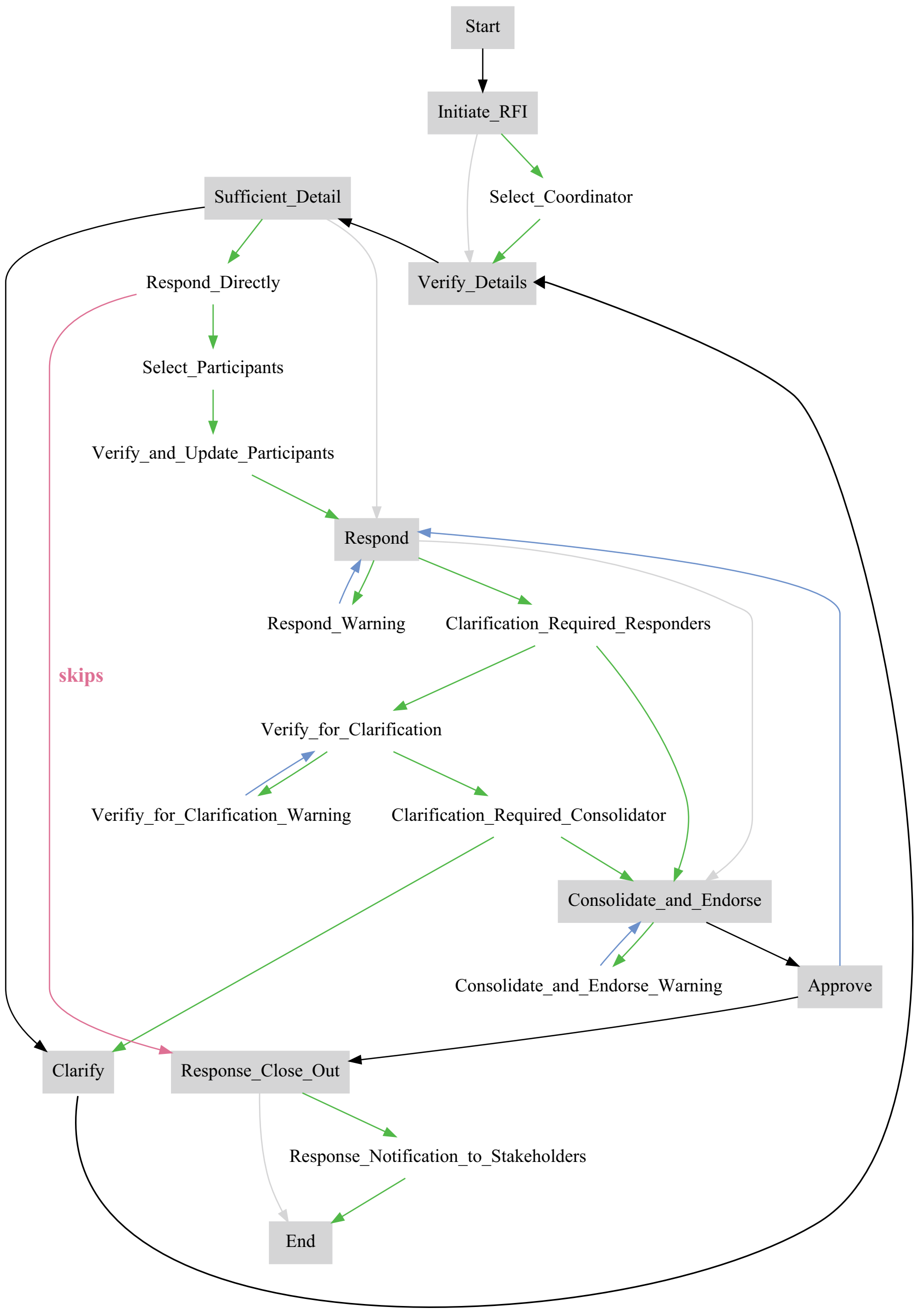


Fig. 10. Overlay analysis of non-conformant customization of the RFI process.

to Alloy format. The Analyzer uses the developed Alloy algorithm to analyze and compare workflows using inheritance rules and determine conformance or non-conformance of the customized workflows compared to the specification workflow. The analysis result is then demonstrated via the Visualizer (see Fig. 11).

8. Limitations and future work

Despite the potential benefits of the Industry Foundation Processes for the construction industry, this study has particular limitations:

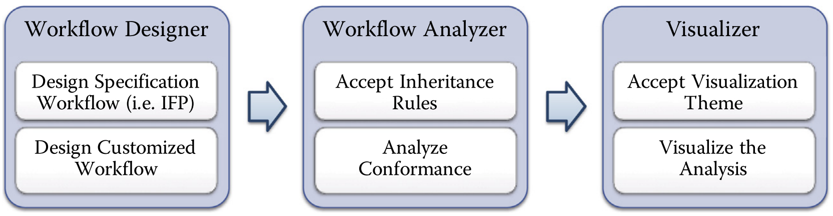


Fig. 11. Three components of the developed conformance checking tool.

1. Although the development approach, implementation tools, and validation methodology for conformance checking of the proposed IFP system have been functionally demonstrated in this paper, a full scale validation via implementation of the IFP system in one or two real projects have not been performed yet, and is considered a future work. Such a full scale implementation and validation would be a better examination and evaluation for the practical benefits of the IFP system within the construction industry.
2. Two main benefits of the IFP initiative are facilitation of conformance and interoperability for complex construction industry workflow processes. The definitions and discussion in this paper are limited to the process conformance details and the automatic conformance checking tool. The IFP interoperability will be discussed in a separate publication.
3. In the construction industry, workflows are often executed in a distributed setting. The prototype implementation presented in this paper has been developed using Microsoft Workflow Foundation technology that fully supports distributed systems, but it has been implemented as a desktop application. Since Workflow Foundation facilitates separation of process design and process enactment from the type of application, and because the same classes that are typically used in distributed systems have been used in this desktop application, the implemented system can be considered an impartial validation for implementation of the RFI process. However, it is still a limitation of this study which can be addressed in future work by developing a web-based distributed system.
4. The IFP system and its ontology components have been defined based on careful investigation and analysis of several process implementations, and consultation with industry experts; however, having access to and analyzing process implementations in more projects might lead to some updates on the definition of components or details.
5. The inheritance rules that have been used for workflow conformance checking in this paper are strict rules that do not allow skipping any of the core activities or changing the sequence of them. As a future work these rules can be relaxed to some extent, *i.e.* to allow change in the sequence of particular core activities, or to allow skipping particular core activities in certain situations, and investigating how these changes affect the conformance checking algorithm.
6. The CAP Theorem [44,45] states that any distributed system may have at most two out of the three of Consistency, Availability, and Partitioning. The prototype presented here requires a distributed system foundation that provides Consistency and Availability, and therefore cannot be Partitioned. Future work could explore different points in this system-design trade-off space.

9. Summary and conclusions

This paper introduces the concept of Industry Foundation Processes (IFP) and provides an ontology for its development and application. IFPs are simple structured processes with the essence of industry best practices. They possess particular features, such as abstraction and inheritance that enable them to systematically be expanded to more complex processes tailored for specific types and conditions of construction projects. Explicit workflow inheritance rules not only allow methodical customization of IFP processes, but also enable automated conformance checking of any workflow with its associated IFP process. In addition, an accepted core structure for an IFP process facilitates process interoperability, which will be discussed in more detail in a separate publication.

Furthermore, this paper discusses the workflow inheritance notion and compares it with the traditional programming inheritance concept. It clarifies that they are different, and both are necessary for implementation of the IFP system. A prototype example of an IFP for the Request for Information (RFI) process – a commonly used process in the construction industry – has been developed in this paper, using C# programming language and Microsoft Workflow Foundation technology, to demonstrate the concept of an IFP system. The concept and methodology introduced, however, can be applied to any other common process in the construction industry, such as risk management, contract management, quality management, and lessons learned.

In addition, automated conformance checking of any workflow with its associated IFP, based on the workflow inheritance rules, has been addressed in detail by developing an algorithm in a first-order logic language. Alloy, a structural modeling language based on first-order logic, developed by the Software Design Group at MIT, is used to compare a customized version of a workflow with its associated IFP. The XAML file of the developed workflow in Visual Studio environment contains the structure of the workflow. This structure is transformed into the format accepted by Alloy to automate the conformance checking process directly from the workflow development environment.

Introducing the theory, application, and potential value of the IFP system is expected to open new research initiatives to enhance process conformance and improve process interoperability in the domain of the construction industry. Developing IFP processes for some of the known processes, such as Change Management, Risk Management, or Lessons Learned, based on industry best practices, and expanding the concept of IFP interoperability in more detail are among the next steps of this research initiative.

While industry partners and experts consulted in this research process highly value the automated conformance checking demonstrated here as a tool for quality assurance and risk management purposes, future research that would compare its deployment on a large set of mega-projects with current workflow management and implementations protocols would be worthwhile for also validating the relationships hypothesized in Fig. 1.

Acknowledgements

This research is funded by the Natural Sciences and Engineering Research Council of Canada (NSERC CRD) and Coreworx Inc. It is also supported by in-kind contributions of Coreworx including technical and expert feedback. These supports are gratefully acknowledged. The authors would like to particularly acknowledge the support of Joel Gray and Paul Harapiak from Coreworx, and the contributions of coop students Ming Zhou for implementation of the RFI process in WF, and Tao Wu for developing a convertor of process models form WF to Alloy and for improving the Alloy analysis.

References

[1] M. Weske, Business Process Management: Concepts, Languages, Architectures, Springer, 2012, http://dx.doi.org/10.1007/978-3-540-73522-9.
[2] W. van der Aalst, Business process management as the "killer app" for petri nets, Softw. Syst. Model. 14 (2) (2014) 685–691, http://dx.doi.org/10.1007/s10270-014-0424-2.
[3] M. El-Mashaleh, W. O'Brien, R. Minchin Jr., Firm performance and information technology utilization in the construction industry, J. Constr. Eng. Manage. 132 (5) (2006) 499–507, http://dx.doi.org/10.1061/(ASCE)0733-9364(2006)132:5(499).
[4] Y. Kang, W. O'Brien, J. Dai, S. Mulva, S. Thomas, R. Chapman, D. Butry, Interaction effects of information technologies and best practices on construction project performance, J. Constr. Eng. Manage. 139 (4) (2013) 361–371, http://dx.doi.org/10.1061/(ASCE)CO.1943-7862.0000627.
[5] Y. Kang, W. O'Brien, S. Thomas, R. Chapman, Impact of information technologies on performance: cross study comparison, J. Constr. Eng. Manage. 134 (11) (2008) 852–863, http://dx.doi.org/10.1061/(ASCE)0733-9364(2008)134:11(852).

[6] S.-H. Lee, S. Thomas, C. Macken, R. Chapman, R. Tucker, I. Kim, Economic value of combined best practice use, J. Manage. Eng. 21 (3) (2005) 118–124, http://dx.doi.org/10.1061/(ASCE)0742-597X(2005)21:3(118).

[7] Y. Shan, P. Goodrum, D. Zhai, C. Haas, C. Caldas, The impact of management practices on mechanical construction productivity, Constr. Manage. Econom. 29 (3) (2011) 305–316, http://dx.doi.org/10.1080/01446193.2010.538070.

[8] D. Zhai, P. Goodrum, C. Haas, C. Caldas, Relationship between automation and integration of construction information systems and labor productivity, J. Constr. Eng. Manage. 135 (8) (2009) 746–753, http://dx.doi.org/10.1061/(ASCE)CO.1943-7862.0000024.

[9] S. Thomas, S.-H. Lee, J. Spencer, R. Tucker, R. Chapman, Impacts of design/information technology on project outcomes, J. Constr. Eng. Manage. 130 (4) (2004) 586–597, http://dx.doi.org/10.1061/(ASCE)0733-9364(2004)130:4(586).

[10] Y. Kang, W. O'Brien, S. Mulva, Value of it: indirect impact of it on construction project performance via best practices, Autom. Constr. 35 (2013) 383–396, http://dx.doi.org/10.1016/j.autcon.2013.05.011.

[11] M. Skibniewski, S. Ghosh, Determination of key performance indicators with enterprise resource planning systems in engineering construction firms, J. Constr. Eng. Manage. 135 (10) (2009) 965–978, http://dx.doi.org/10.1061/(ASCE)0733-9364(2009)135:10(965).

[12] J. O'Connor, S. Dodd, Achieving integration on capital projects with enterprise resource planning systems, Autom. Constr. 9 (5) (2000) 515–524, http://dx.doi.org/10.1016/S0926-5805(00)00062-5.

[13] B. Chung, M. Skibniewski, Y. Kwak, Developing ERP systems success model for the construction industry, J. Constr. Eng. Manage. 135 (3) (2009) 207–216, http://dx.doi.org/10.1061/(ASCE)0733-9364(2009)135:3(207).

[14] Ghosh S, Negahban S, Kwak Y, Skibniewski M. Impact of sustainability on integration and interoperability between BIM and ERP – a governance framework. In: Technology management conference (ITMC), 2011 IEEE international, 2011, pp. 187–193. http://dx.doi.org/10.1109/ITMC.2011.5995975.

[15] W. Van Der Aalst, Business process management demystified: a tutorial on models, systems and standards for workflow management, Lect. Notes Comput. Sci. 3098 (2004) 1–65 (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics).

[16] J. Cardoso, R.P. Bostrom, A. Sheth, Workflow, Inform. Technol. Manage. 5 (3–4) (2004) 319–338, http://dx.doi.org/10.1023/B:ITEM.0000031584.14039.99.

[17] P. Tang, B. Akinci, Automatic execution of workflows on laser-scanned data for extracting bridge surveying goals, Adv. Eng. Inform. 26 (4) (2012) 889–903, http://dx.doi.org/10.1016/j.aei.2012.07.004.

[18] C. Caldas, L. Soibelman, L. Gasser, Methodology for the integration of project documents in model-based information systems, J. Comput. Civ. Eng. 19 (1) (2005) 25–33, http://dx.doi.org/10.1061/(ASCE)0887-3801(2005)19:1(25).

[19] M. Al Qady, A. Kandil, Document discourse for managing construction project documents, J. Comput. Civ. Eng. 27 (5) (2013) 466–475, http://dx.doi.org/10.1061/(ASCE)CP.1943-5487.0000201.

[20] N. El-Gohary, T. El-Diraby, Dynamic knowledge-based process integration portal for collaborative construction, J. Constr. Eng. Manage. 136 (3) (2010) 316–328, http://dx.doi.org/10.1061/(ASCE)CO.1943-7862.0000147.

[21] Y. Kang, W. O'Brien, J. O'Connor, Analysis of information integration benefit drivers and implementation hindrances, Autom. Constr. 22 (2012) 277–289, http://dx.doi.org/10.1016/j.autcon.2011.09.003.

[22] A. Shahi, C. Haas, J. West, B. Akinci, Workflow-based construction research data management and dissemination, J. Comput. Civ. Eng. 28 (2) (2014) 244–252, http://dx.doi.org/10.1061/(ASCE)CP.1943-5487.0000251.

[23] S. Shokri, M. Safa, C.T. Haas, R.C. Haas, K. Maloney, S. MacGillivray, Interface management model for mega capital projects, in: Construction Research Congress, 2012, pp. 447–456. http://dx.doi.org/10.1061/9780784412329.045.

[24] F. Mannhardt, M. de Leoni, H. Reijers, W. van der Aalst, Balanced multi-perspective checking of process conformance, Computing, http://dx.doi.org/10.1007/s00607-015-0441-1.

[25] E. Ramezani Taghiabadi, D. Fahland, B. Van Dongen, W. Van Der Aalst, Diagnostic information for compliance checking of temporal compliance requirements, Lect. Notes Comput. Sci. 7908 LNCS (2013) 304–320, http://dx.doi.org/10.1007/978-3-642-38709-8_20 (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics).

[26] T.W. Malone, K. Crowston, G.A. Herman, Organizing Business Knowledge: The MIT Process Handbook, MIT Press, 2003.

[27] W. Van der Aalst, T. Basten, Inheritance of workflows: an approach to tackling problems related to change, Theor. Comput. Sci. 270 (1–2) (2002) 125–203, http://dx.doi.org/10.1016/S0304-3975(00)00321-2.

[28] W. Van Der Aalst, Inheritance of business processes: a journey visiting four notorious problems, Lect. Notes Comput. Sci. 2472 (2003) 383–408, http://dx.doi.org/10.1007/978-3-540-40022-6_19 (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics).

[29] W.M. van der Aalst, Inheritance of dynamic behaviour in UML, MOCA 2 (2002) 105–120, http://dx.doi.org/10.1016/S1567-8326(00)00004-7.

[30] D. Hollingsworth, Workflow Handbook 1997, John Wiley & Sons, Inc., New York, NY, USA, 1997.

[31] B. White, Pro WF 4.5, Apress, 2012, http://dx.doi.org/10.1007/978-1-4302-4384-7.

[32] Microsoft Developer Network, The workflow way: Understanding windows workflow foundation. <https://msdn.microsoft.com/en-us/library/dd851337.aspx> (accessed 21.04.2015).

[33] Microsoft Developer Network, A developer's introduction to windows workflow foundation (wf) in.net 4. <https://msdn.microsoft.com/en-us/library/ee342461.aspx> (accessed 21.04.2015).

[34] Microsoft Developer Network, State machine workflows. <https://msdn.microsoft.com/en-us/library/ee264171(v=vs.110).aspx> (accessed 21.04.2015).

[35] R.T. Prosser, Applications of boolean matrices to the analysis of flow diagrams, in: IRE-AIEE-ACM Eastern Joint Computer Conference, ACM, Boston, MA, USA, 1959, pp. 133–138.

[36] T. Lengauer, R.E. Tarjan, A fast algorithm for finding dominators in a flowgraph, ACM Trans. Program. Lang. Syst. 1 (1) (1979) 121–141, http://dx.doi.org/10.1145/357062.357071.

[37] L. Georgiadis, R.F. Werneck, R.E. Tarjan, S. Triantafyllis, D.I. August, Finding dominators in practice, J. Graph Algorithms Appl. 10 (1) (2006) 69–94.

[38] D. Jackson, Software Abstractions: Logic, Language, and Analysis, MIT Press, 2012, http://dx.doi.org/10.1017/S0956796808006977.

[39] J.T. Schwartz, R.B.K. Dewar, E. Dubinsky, E. Schonberg, Programming with Sets: An Introduction to SETL, Springer Verlag, NY, USA, 1986.

[40] S.A. Cook, The complexity of theorem-proving procedures, in: Proceedings of the Third Annual ACM Symposium on Theory of Computing, STOC '71, ACM, New York, NY, USA, 1971, pp. 151–158, http://dx.doi.org/10.1145/800157.805047.

[41] N. En, N. Srensson, An extensible SAT-solver, Lect. Notes Comput. Sci. 2919 (2004) 502–518, http://dx.doi.org/10.1007/978-3-540-24605-3_37 (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics).

[42] G. Dennis, A relational framework for bounded program verification, Ph.D. Thesis, MIT EECS, 2009.

[43] J.R. Ullmann, An algorithm for subgraph isomorphism, J. ACM 23 (1) (1976) 31–42.

[44] A. Fox, E. Brewer, Harvest, yield and scalable tolerant systems, in: Proc. 7th Workshop Hot Topics in Operating Systems (HotOS 99), IEEE CS, 1999, pp. 174–178.

[45] S. Gilbert, N. Lynch, Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services, ACM SIGACT News 33 (2) (2002) 51–59.