Modelling and implementation of correct by construction healthcare workflows

Modelling and implementation of correct by construction healthcare workflows

Petros Papapanagiotou and Jacques Fleuriot

School of Informatics, University of Edinburgh,
10 Crichton street, Edinburgh EH8 9AB, United Kingdom
ppapapan@inf.ed.ac.uk,jdf@inf.ed.ac.uk

Abstract. We present a rigorous methodology for the modelling and implementation of correct by construction healthcare workflows. It relies on the theoretical concept of proofs-as-processes that draws a connection between logical proofs and process workflows. Based on this, our methodology offers an increased level of trust through mathematical guarantees of correctness for the constructed workflows, including type correctness, systematic resource management, and deadlock and livelock freedom. Workflows are modelled as compositions of abstract processes and can be deployed as executable code automatically. We demonstrate the benefits of our approach through a prototype system involving workflows for assignment and delegation of clinical services while tracking responsibility and accountability explicitly.

Keywords: process modelling in healthcare; formal verification; workflow automation; healthcare process integration

1 Introduction

The primary aim of our research is to combine and use the rich theory of proofs-as-processes [1] and a rigorous, logical engine to develop a pragmatic methodology for the development of trustworthy, correct by construction healthcare workflows. This allows the combination of the benefits from both Business Process Modelling (BPM) and formal methods, so that in addition to the flexibility, scalability, maintainability, and separation of concerns offered by process modelling, we can obtain an added level of trust of the correctness and consistency with regards to the modelled workflows.

In particular, our efforts focus on the management of information and resources in clinical and administrative procedures involving health and social care providers. We provide a framework that allows a high level modelling of such procedures that can lead to the reduction of redundancies and process repetitions, better enforcement of policies and continuity and consistency of practice, while abstracting from the complex clinical decision making. Ultimately, this could improve patient safety and reduce costs and the time spent by carers in their effort to adhere to guidelines.

Our methodology allows the construction of workflows as process compositions. Component processes can be specified abstractly based on their inputs, outputs, preconditions and effects. Our framework offers mathematical guarantees of correctness with respect to the information flow, resource management, typing, and deadlock and livelock freedom. Most importantly, our correct by construction workflows can be automatically deployed as executable code.

We begin the analysis of our approach by describing the core theoretical background in Section 2, followed by a breakdown of the methodology in Section 3. We analyse the added benefits of our formal approach in Section 4, whereas Section 5 describes the kind of healthcare processes that our approach is tailored to. We then demonstrate our methodology in a practical healthcare application, with emphasis on the deployment stage, in Section 6. We conclude with an overview of lessons learned with respect to the application of our methodology in healthcare in Section 7, brief comparison to related work in Section 8, and our plans for future work in Section 9.

2 Theoretical background

Our approach involves validating that a system is correct in the sense that it is mathematically guaranteed to give the expected result based on its specification. Such guarantees of correctness are particularly important in healthcare systems, where maintaining safety and policy adherence is crucial and a bug in the code may have severe implications to patients' lives. For the purpose of formally verifying healthcare workflows, as already mentioned, we make use of the proofs-as-processes paradigm, which involves a mapping between logical proofs and processes described in process calculus terms.

Essentially, we can construct logical specifications of processes and process workflows using Classical Linear Logic (CLL) [5], which emphasizes formulas that represent resources. Assumptions cannot be ignored or copied so no resources can duplicate or vanish. CLL allows the construction of logical specifications of processes with respect to the types of their inputs, outputs, preconditions, and effects (IOPEs). Moreover, CLL allows the specification of IOPEs that are either parallel (simultaneous) or optional. Optional IOPEs can be used, for example, to express the possibility of a process throwing an exception instead of producing the expected result (e.g. when an unexpected obstacle occurs during an operation). It is worth mentioning that most process composition methodologies do not give explicit considerations to exceptions.

In order to abstract from the complicated underlying CLL specifications, we have devised an intuitive, diagrammatic representation [15], where processes are boxes with solid edges for parallel IOPEs and dashed edges for optional ones.

For instance, consider a possible specification for the process describing a blood test. In order to perform the test, we require the patient details, a referral, a scheduled time, and a blood sample. The blood test results can be either conclusive or inconclusive (in which *exceptional* case there may be a decision for

repeating the blood test at a later date). This process can be specified in CLL and represented diagrammatically as shown in Figure 1.

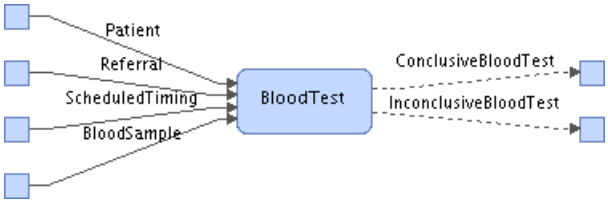

Fig. 1. Diagrammatic specification of the blood test process.

Using the proofs-as-processes paradigm, CLL workflow specifications can be translated to executable terms in the π-calculus [10], which is a formalism aimed at the description of concurrent processes as independent, atomic entities. These communicate asynchronously by message passing. Over the years, the π-calculus has inspired a variety of process algebras as well as BPM languages such as BPEL, and has been used as the means to formalise their semantics [8].

The combination of CLL and the π-calculus forms the basis for a formal semantics and rigorous workflow design, thus leading to workflow systems that are *correct by construction*, based on the methodology described next.

3 Methodology

Our methodology has been developed within the following set of assumptions:

1. We assume a set of atomic healthcare processes that can interface a variety of services, including Electronic Medical Records (EMRs), medical equipment and instruments, and Human Provided Services (HPSs) through the use of electronic forms. Each of these can be described using a type specification of their inputs, outputs, preconditions, and effects (IOPEs).
2. The methodology is agnostic to the inner working of the available processes, which are treated as 'black boxes'. We assume the processes are well behaved and always satisfy their type specification.
3. We assume all of the available processes always terminate.

Based on these assumptions, our methodology follows a standard process modelling approach. We consult with a variety of stakeholders, including healthcare practitioners, administrators, policy makers, and patients to breakdown the selected task into a number of individual but interdependent steps. The consultation is most commonly in the form of contextual interviews, shadowing, questionnaires, and frequent communication.

We then proceed to construct specifications of these steps as processes by identifying their IOPEs and formalising them using our logic based system. The

system allows the combination of these processes to construct composite workflows. More specifically, processes can be combined in sequence, in parallel, and conditionally. Each process combination is formally verified by our underlying logic engine which performs the necessary inference steps automatically.

Once the composition stage is complete, the system provides a complete, executable, concurrent π-calculus specification for the constructed workflow. This allows the user to perform visual simulations using a built-in tool and empirically verify the behaviour of the workflow before deploying it as a live system.

Our system also provides advanced deployment functionality that allows the user to export executable Scala [13] code for a constructed workflow automatically. This is accomplished through our implemented extension of Scala's PiLib library [3] which allows a direct translation of π-calculus terms into Scala code.

The roadmap for our deployment procedure is shown in Figure 2. Since we assume available processes are 'black boxes', the deployed code represents those as partially abstract classes (also known as *traits*) which need to be implemented as a concrete instance. The code of the Scala class responsible for the workflow execution and associated information flow is generated automatically.

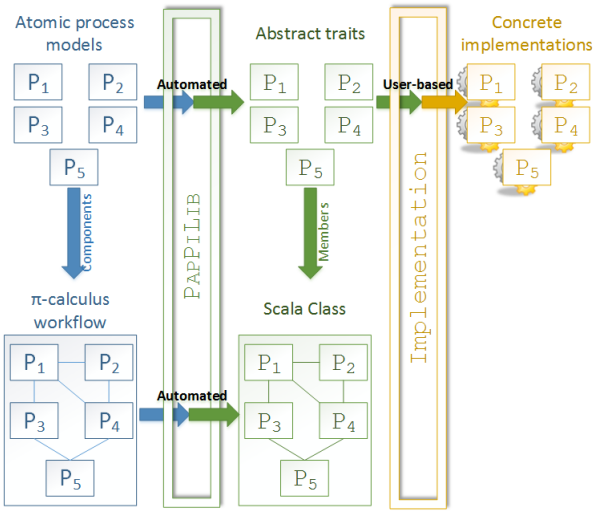

Fig. 2. The general roadmap of the workflow deployment procedure.

Scala allows a seamless integration with any available Java library, which makes the deployed system flexible with respect to integrating modern tools, such as EMRs, e-form technologies, healthcare devices etc.

The deployed code coordinates the available processes according to the corresponding formally verified workflow. It executes concurrently and asynchronously while remaining free of deadlocks and livelocks.

4 Benefits through proofs-as-processes

The mathematical core of the proofs-as-processes approach combines the properties of CLL with the concurrent computation of the π-calculus. We have embedded this theory in a logic-based engine called HOL Light [7]. The resulting framework allows the user to perform formally verified inference and construct process models, based on the methodology described above, while providing the following unique benefits with respect to the stated assumptions:

1. **Explicit, verified information/resource flow:** The constructed models aim to make the information and resource flow between existing atomic processes (Assumption 1) explicit and consistent. This includes enforcing the resource dependencies between the processes as introduced by the user, and allowing fine grained control of the process execution order as well as which processes should execute in which cases. This minimizes unnecessary clinical procedures and guarantees that all the necessary information (test results, patient records, clinical assessments, etc.) and resources (drugs, samples, equipment, etc.) will be available before any procedure is initiated.
2. **Systematic resource management:** As already mentioned, the properties of CLL disallow any *implicit* duplication or consumption (vanishing) of resources. This greatly facilitates resource accounting during composition so that the user does not need to manually keep track of resources, especially when there is a large number of them at a given stage. Limited resources, such as reusable results from costly, lengthy, or invasive medical tests, can not "magically" appear or disappear. Moreover, all possible outcomes (including exceptions that are often forgotten about), as defined by the IOPEs of the involved processes (Assumption 1) must be handled explicitly.
3. **Concurrent execution and deadlock and livelock freedom:** The composed workflow is executed concurrently in order to maximize efficiency. This allows for independent clinical procedures to be performed simultaneously, thus saving time, without the user having to explicitly state this (unless there is an explicit or implicit dependency, processes will run in parallel by default). At the same time, if each component process always terminates (Assumption 3), the theoretical background of our methodology guarantees that termination is preserved in the composed workflow and no deadlocks or livelocks are introduced during composition. This prevents, for example, the workflow being blocked by 2 clinicians waiting for feedback from each other.
4. **Type correctness during composition:** The logical engine guarantees the correct matching of the user-defined types (Assumption 1) as processes are being composed and eventually leads to executable code that is typechecked in advance. This allows deployment in untyped programming languages as well as integration of heterogeneous components. It also provides a degree of consistency and continuity in the workflow in the sense that it guarantees that the involved resource types do not mutate during execution.
5. **Automated workflow deployment:** The result of our composition is immediately translatable to executable code that can be used both for simulation purposes and in a production setting (e.g. a hospital). This minimizes

the time and cost of workflow deployment which is particularly important, especially in cases of maintenance where minor updates in the process models or the corresponding healthcare policies can take a considerable amount of time to be implemented in practice. In our case, this happens with the click of a button. Moreover, the abstraction from the inner working of each process (Assumption 2) allows our workflows to integrate with existing technologies, including medical devices, EMRs, mobile devices etc.

It is worth mentioning that our diagrammatic interface [15] hides the underlying reasoning engine so that little to no logic expertise is required to use our framework. The user applies mouse gestures to compose processes diagrammatically, with all logical inference steps taken care of automatically in the background. The resulting diagram depicts the information flow in the composite model so that it can be understood by a variety of stakeholders, including health carers, policy makers, and IT developers (see Figure 3 for an example).

5 Healthcare Processes

The approach is tailored towards the modelling of everyday practices of care providers. These are governed by finite processes that repeat for every individual case, but are most commonly informal and require considerable amounts of time and effort from the carers, including the effort to track individual patient pathways daily by memory.

As previously mentioned, we focus on the exchange of resources and information in such processes, and we aim to provide automated coordination through verified workflow modelling. We, therefore, model individual, finite, terminating processes (that exclude iterative or feedback processes), which may involve the provision of healthcare services, administrative or documentation procedures, as well as interfacing with electronic systems such as medical devices and EMRs.

More specifically, our CLL based formalism, caters for abstract, high level process specifications and does not allow loops (but can include finite iterations). Although this level of expressivity may appear relatively simple in comparison to other workflow languages, and, moreover, introducing deadlock and livelock freedom in a language without loops may be viewed as less challenging, our recent experience in real-world healthcare modelling indicates that our framework fits well within the current needs of healthcare stakeholders (see Section 7).

We proceed to demonstrate our methodology in more detail through a working example involving collaboration patterns in healthcare.

6 Formal verification of collaboration patterns in healthcare

In our primary example, we use our formally verified process modelling methodology to model and deploy patterns of collaborative work in healthcare, originally described in recent work by Grando et al. [6]. Modelling aspects of this work have

been described in detail in previous papers [14, 16], whereas in the current work we concentrate on workflow automation.

We focus our investigation on basic collaboration scenarios in healthcare that involve two agents, also known as *actors*. These may correspond to any member of the medical staff, including doctors and nurses. We are particularly interested in two patterns or skeletal plans that differentiate between the types of collaboration through *assignment* and *delegation* of clinical services. In these, one of the two actors, the *requester*, asks for a particular clinical service (e.g. specialized diagnosis or treatment, administration of a drug, etc.) from the other, the *provider*. In order for a contract to be signed between the requester and the provider, it must be ensured that the provider is competent to perform the service. Moreover, depending on whether the service is assigned or delegated, responsibility and accountability is either transferred to the provider (assignment) or maintained by the requester (delegation). We proceed to explain the approach in more detail in the next few sections.

6.1 Process modelling

The first stage of our BPM-inspired methodology involves the breakdown of the individual steps. Clinical services are broken down to a series of individual tasks and goals, also known as *keystones*, that each of the actors must perform in order to complete the service. These include tasks that must be performed in exceptional cases and unexpected situations. Each keystone has preconditions that must be met for it to be achieved, and success conditions that describe the achieved effects upon its successful completion. We give a brief description of the modelled keystones for the our healthcare collaboration patterns next:

- The `ServiceASSGRequest` and `ServiceDELGRequest` keystones initiate a request for an assignment and a delegation of a patient respectively.
- The `CollabDecision` keystone corresponds to the decision of whether there is a competent actor available to provide the service of a requested contract.
- Once a contract has been accepted by a competent provider the requester can finalize the agreement in the `ContractAwarded` keystone.
- The `ServiceProvide` keystone corresponds to the task of executing a requested clinical service that is currently pending. The service may either be completed successfully or an obstacle may occur preventing its completion.
- The `OutcomeCheck` keystone corresponds to the goal of checking the outcome of the provided service by the responsible actor. For simplicity we assume this goal is always successful.
- The `AssgResponsible` and `DelegResponsible` keystones correspond to automated procedures that determine the responsible actor in the cases of an assignment and a delegation respectively.

We proceed to compose these keystones/processes in 2 complete workflows that fully describe the patterns of assignment and delegation and enforce the responsibility and accountability constraints. Due to space limitations we only present the assignment workflow in Figure 3, although, for those interested, both diagrams can be found in a previous paper [16].

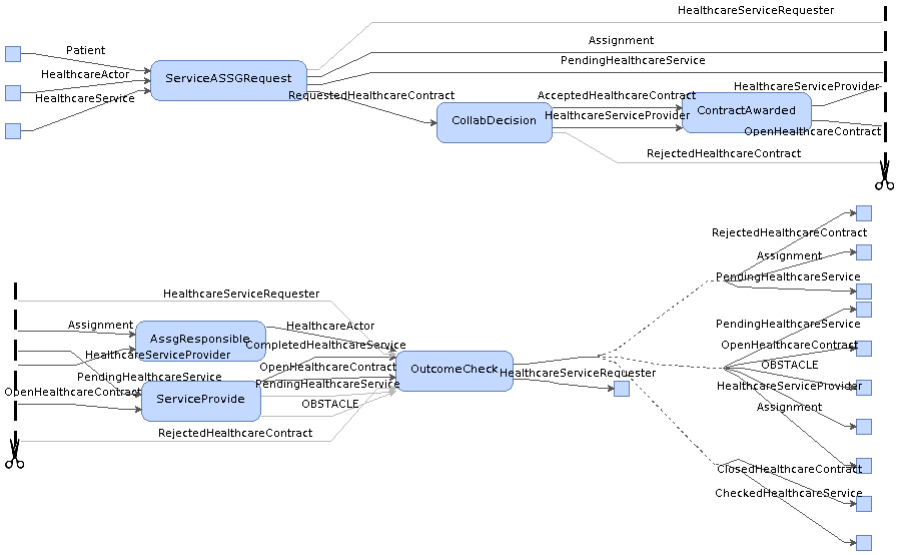

Fig. 3. Diagram of the *assignment* pattern as a verified process composition. Note that the pattern has been split into 2 parts at the indicated point for it to fit within the page.

6.2 Workflow automation

We now come to the crux of the current paper: once the healthcare collaboration patterns have been modelled as formally verified workflows using our logic-based framework, our system allows the automatic deployment of Scala code in order to achieve computer-based coordination of the assignment and delegation procedures. The only requisite in order to produce a fully functioning system is the implementation of the keystones/processes, which can then be integrated directly in the deployed coordinator.

In order to demonstrate the functionality of such a deployment, we have developed a web-based system that emulates part of a hospital environment. We will simply refer to this system as the *DigiHealth* (Digital Healthcare) prototype.

DigiHealth includes a minimal relational database with dummy data about patients, medical staff, clinical services, handover contracts, and potential obstacles. Each keystone/process is implemented as an interaction with *DigiHealth*'s API. The order in which the API calls (process invocations) are made and the involved information is exchanged are both dictated by the formally verified workflow as opposed to some hardcoded solution. In this way, we obtain a full transition from a information/resource flow diagram (see Figure 3) to the coordination of actual administrative procedures for the assignment and delegation of patients, with *all* the properties provided by our logic-based framework (see Section 4). Additional workflows can be defined on top of the same available pro-

cesses, so that different guidelines, cf. the patient delegation case in the previous section, can be implemented while reusing the existing process implementations.

The user interface is accessible by any mobile device, and provides all the necessary information to the user, including available contracts, pending requests, pending services, and the ability to initiate new assignments or delegations of patients. A sample screenshot of the interface is shown in Figure 4.

It is worth noting that the user is only shown information and processes that are relevant to them. They are relieved from the burden of keeping track of patients, monitoring the state of the assignment or delegation taking place (or more generally the state of the patient flow), coordinating the next steps, or ensuring the information is communicated to all the necessary clinical collaborators. Using simple selections, e-form based point and clicks, and standard gestures, the users only perform their own part, while the the deployed coordinator takes care of the workflow based on the diagrammatic, logic-based specification.

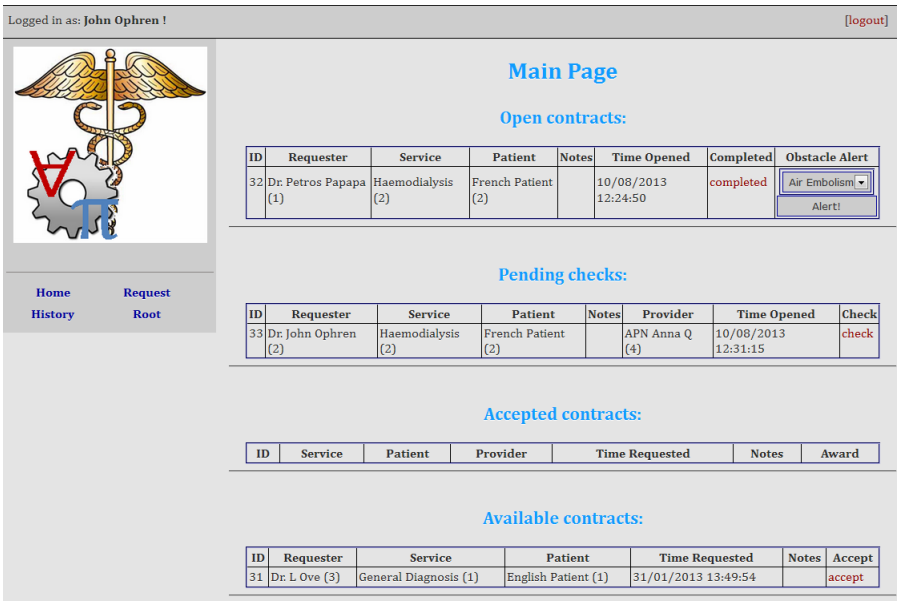

Fig. 4. Sample screenshot from the prototype DigiHealth system.

7 Lessons learned

Our formally verified process modelling is currently being applied to real-world projects involving (distinct groups of) UK clinicians interested in aspects such intra-hospital patient transfers and integrated care pathways. For example, we are collaborating with leading HIV clinicians in Glasgow and Edinburgh in an ongoing effort for the automation of Integrated Care Pathways for HIV patients in Scotland. Our preliminary models include the first three months of care for

HIV patients, including assessment and consultation. Based on these collaborations, we believe that our methodology is particularly applicable to healthcare for the following reasons:

1. In real world healthcare scenarios, there is a very large number of resources being exchanged at every stage as well as many conditional and exceptional situations, and it often becomes cumbersome to keep track of or record them in a formalised healthcare procedure. Our framework greatly facilitates this task, thanks to its resource accounting mechanisms.
2. The task of tracking information and resources itself is deemed of high importance in healthcare, because it affects the efficiency, time, and cost of every day care provision and, more importantly, patient safety, since, for example, patient drop outs due to neglected pathways are common.
3. It is often challenging to communicate the technicalities of workflows to healthcare practitioners. Experience has shown that our diagrammatic models are easily understandable by clinical collaborators and allow them to reflect upon their practices. The prospect of automated deployment is also very attractive to carers, who spend a large fraction of their valuable time tracking, memorizing, and documenting pathways.
4. The mathematical core of our framework provides guarantees that enable all stakeholders to trust that the modelled workflow will behave correctly.
5. The formal underpinnings of the used languages, namely CLL and the π-calculus, in combination with the abstract level of the process specifications open the doors to further analysis and verification, including the possibility to integrate with other workflow, simulation, and verification technologies.

To summarize, despite the relative simplicity of our process specifications, our methodology seems to fit well for the particular purpose of formalising healthcare procedures such as integrated care pathways as workflows to improve every day practices of healthcare providers.

8 Related Work

The current research is inspired by recent work by Grando et al. [6]. In this, they proposed logic-based, pen-and-paper specifications of reusable patterns for specifying assignment and delegation of tasks and goals during collaborative work and sketched proofs that desirable properties, known as *safety principles*, related to accountability, responsibility, and competence could be ensured.

Workflow-based approaches are often used in healthcare informatics to provide automated IT support for practitioners. Tallis [17], for example, is one of the leading tools for the specification and enactment of clinical applications. Such tools are most commonly used in an effort to support decision making in diagnosis and treatment. In our case, though, we are closer to a business process oriented approach that abstracts from these procedures and focuses on improving the operational support and automating some of the organizational aspects of patient care by, for instance, alleviating the need to communicate the same

information to multiple people separately, automatically documenting repetitive information, and detaching social conventions from the healthcare workflow.

A related approach focusing on organisational workflows in healthcare with the aim of minimizing medical errors is presented by Malhotra et al. [9]. They focus on building a cognitive model of the workflow in order to identify error-prone regions. TESTMED [2] is another related project aimed towards providing operational support in hospital wards through multimodal interfaces.

The main advantage of our approach compared to others is the formal verification of the constructed patterns and the automated extraction of an executable model. The end-product is a system whose correctness is mechanically verified with associated guarantees regarding the enforcement of modelled conditions and policies, thereby allowing a high level of trust in the implemented workflow.

9 Conclusion and Future work

In this work, we presented a formal verification approach to the modelling and implementation of healthcare workflows. Our methodology takes advantage of the proofs-as-processes paradigm, a theory that connects logical proofs with process workflows, in order to generate correct by construction process compositions. In particular, we provide mathematical guarantees of properties for the generated workflows, including a verified information flow, systematic resource management, type correctness, and deadlock and livelock freedom. Workflows are modelled as compositions of abstract processes, specified by their inputs, outputs, preconditions, and effects. The verified models can be automatically deployed as executable Scala code, thus greatly facilitating workflow automation and maintenance. The deployed systems support integration and interoperability with existing infrastructure and may be accessed by mobile devices.

We demonstrate the value our methodology adds to traditional process models through the prototype system *DigiHealth*. Patterns of collaboration in healthcare involving *assignment* and *delegation* of clinical services are formalised and deployed on *DigiHealth*, allowing for the automation of the corresponding procedures based on a verified protocol. The stakeholders can work their way through the two workflows using the web interface, without the need to keep track of the progress themselves, and with the added trust our rigorous framework provides.

Our plans for future work involve a variety of optimisations on the generated code, including distributed execution (for deployment on the Cloud) and tracking of workflow analytics. In addition, we plan to explore connections and mappings to existing technologies such as the Business Process Model and Notation (BPMN) [12], the Business Process Execution Language (BPEL) [11], and RESTful web services [4]. Finally, we are looking to pursue further collaborations with healthcare providers and policy makers to investigate other potential uses of our methodology for the formalisation of healthcare procedures.

Acknowledgments. This research was supported by an EPSRC doctoral scholarship, by EPSRC grant EP/J001058/1, and by a grant from the College of Sci-

ences and Engineering of the University of Edinburgh. We would like to thank the reviewers for their constructive comments.

References

1. Bellin, G., Scott, P.: On the π-calculus and linear logic. Theoretical Computer Science 135(1), 11–65 (1994)
2. Cossu, F., Marrella, A., Mecella, M., Russo, A., Bertazzoni, G., Suppa, M., Grasso, F.: Improving operational support in hospital wards through vocal interfaces and process-awareness. In: Computer-Based Medical Systems (CBMS), 2012 25th International Symposium on. pp. 1–6. IEEE (2012)
3. Cremet, V., Odersky, M.: PiLib: A hosted language for Pi-Calculus style concurrency. In: Lengauer, C., Batory, D., Consel, C., Odersky, M. (eds.) Domain-Specific Program Generation, Lecture Notes in Computer Science, vol. 3016, pp. 180–195. Springer Berlin Heidelberg (2004), http://dx.doi.org/10.1007/978-3-540-25935-0_11
4. Elkstein, M.: Learn rest: A tutorial (February 2008), http://rest.elkstein.org/
5. Girard, J.Y.: Linear logic: its syntax and semantics. In: Girard, J.Y., Lafont, Y., Regnier, L. (eds.) Advances in Linear Logic. No. 222 in London Mathematical Society Lecture Notes Series, Cambridge University Press (1995), http://iml.univ-mrs.fr/~girard/Synsem.pdf.gz
6. Grando, M.A., Peleg, M., Cuggia, M., Glasspool, D.: Patterns for collaborative work in health care teams. AI in Medicine 53(3), 139–160 (2011)
7. Harrison, J.: HOL Light: A tutorial introduction. Lecture Notes in Computer Science pp. 265–269 (1996)
8. Lucchi, R., Mazzara, M.: A pi-calculus based semantics for ws-bpel. The Journal of logic and algebraic programming 70(1), 96–118 (2007)
9. Malhotra, S., Jordan, D., Shortliffe, E., Patel, V.L.: Workflow modeling in critical care: Piecing together your own puzzle. J. of Biomedical Informatics 40(2), 81–92 (Apr 2007), http://dx.doi.org/10.1016/j.jbi.2006.06.002
10. Milner, R.: Communicating and mobile systems: the π-calculus. Cambridge Univ Presss (1999)
11. OASIS: Web Services Business Process Execution Language, version 2.0, OASIS Standard (2007), http://docs.oasis-open.org/wsbpel/2.0/OS/
12. Object Management Group: Business Process Model and Notation (BPMN), version 2.0 (2011), http://www.omg.org/spec/BPMN/2.0/PDF
13. Odersky, M.: The Scala language specification, version 2.8. Programming Methods Laboratory, EPFL Lausanne, Switzerland (October 2013)
14. Papapanagiotou, P., Fleuriot, J., Grando, A.: Rigorous process-based modelling of patterns for collaborative work in healthcare teams. In: Computer-Based Medical Systems (CBMS), 2012 25th International Symposium on. pp. 1–6. IEEE (2012)
15. Papapanagiotou, P., Fleuriot, J., Wilson, S.: Diagrammatically-driven formal verification of web-services composition. In: Cox, P., Plimmer, B., Rodgers, P. (eds.) Diagrammatic Representation and Inference, Lecture Notes in Computer Science, vol. 7352, pp. 241–255. Springer Berlin Heidelberg (2012), http://dx.doi.org/10.1007/978-3-642-31223-6_25
16. Papapanagiotou, P., Fleuriot, J.D.: Formal verification of collaboration patterns in healthcare. Behaviour & Information Technology (2013)
17. Tallis: The tallis toolset (2011), http://archive.cossac.org/tallis/