Electronic Communications of the EASST
Volume 37 (2011)

Workshops der wissenschaftlichen Konferenz Kommunikation in verteilten Systemen 2011 (WowKiVS 2011)

Flexible Behaviour of Human Actors in Distributed Workflows

Adwoa Donyina and Reiko Heckel

12 pages

Guest Editors: Horst Hellbrück, Norbert Luttenberger, Volker Turau
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Flexible Behaviour of Human Actors in Distributed Workflows

Adwoa Donyina[1] **and Reiko Heckel**[2]

[1] add7@le.ac.uk
http://www.cs.le.ac.uk/~add7
Department of Computer Science, University of Leicester, Leicester, United Kingdom
[2] reiko@mcs.le.ac.uk
http://www.cs.le.ac.uk/~rh122
Department of Computer Science, University of Leicester, Leicester, United Kingdom

Abstract:

Workflows often require participation of technical components as well as humans to achieve their objectives. In order to accurately model and simulate the flexibility of human and technical resources we have to consider diverse aspects such as (dynamic) (re)assignment, deadlines, escalation handling, faults and retrieval (backtracking).

In order to meet these requirements we will represent configurations of a business process by graphs and operations changing these configurations by graph transformation rules. We will propose a domain-specific notation to visualise rules and configurations. Adding stochastic time, we can simulate process models to assess their performance.

In a distributed setting, workflows can balance the load between different organisations, thus increasing flexibility, performance and reliability. We will use simulations to compare non-distributed and distributed versions of a system realising workflows in dynamic commercial environment. We will illustrate and validate the approach by means of a pharmacy case study.

Keywords: stochastic modelling and simulation; graph transformation; organisational metamodelling; business processes; escalation handling; dynamic role (re)-assignment; distributed systems; load balancing

1 Introduction

Businesses need a means to test scheduling protocols, policies and regulations, prior to employing them in their day-to-day operation. With the goal of increasing business productivity, simulations can provide an analysis facility for metrics measuring service levels achieved by different policies. We will use a simple graphical syntax targeting business experts to precisely describe activities and policies.

It is difficult to accurately model dynamic role allocation, because in contrast to computer systems, human behaviour is only predictable to a certain degree of probability. In semi-automated business processes human actors are guided by predetermined policies and regulations, but retain the freedom to react to unforeseen events, or maliciously disregard procedures. Hence, in

order to accurately capture this aspect of human behaviour we considered a variety of issues such as escalation handling, load balancing, access control, assignment policies, and scheduling protocols. The scope of this paper will focus on the following four key modelling requirements:

R1 *Dynamic (re)-allocation of roles*: Within distributed systems, tasks can be dynamically (re)assigned locally to people based on their availability and capabilities.

R2 *Temporal Escalation handling*: Deadlines can trigger escalations, which lead to resources being diverted to the escalated case. Procedures should be in place as to how escalations are handled to comply with legal requirements and allowed to react efficiently.

R3 *Scheduling and Load Balancing* : Scheduling and load balancing should maximise throughput and minimise the number of tasks completed past the deadline. Load balancing allows for dynamically (re)assigning tasks globally to other sites (locations).

R4 *Human Error and Recovery (Backtracking)*: Human error may increase the completion time of a task because additional backtracking is required in order to eliminate errors. We also need to keep in mind that human beings are autonomous and even if people are correctly instructed, they may or may not perform their allocated tasks.

Task allocation refers to the breakdown of the total workload into smaller activities to be assigned to different processors [Ze96]. In order to increase performance, it is important that each processor is properly utilised. Hence scheduling and load balancing between resources will tend to increase organisations' performance.

We will discuss how existing methods fail to satisfy these four requirements in more detail in Section 3. Other requirements such as access control, process scheduling, role promotion and demotion were further discussed in [Don10].

To manage the complexity of such organisational models we require the support of a dedicated language with domain-specific notation and formal support for analysis. Classical workflow modelling languages based on control-flow oriented or net-like diagrams are often not flexible enough to allow for dynamic load balancing, while more comprehensive approaches lack the formal semantics and analysis capabilities. Our approach is aimed at human actors in business processes, replacing the rigid control flow with a flexible rule-based approach. The domain-specific language incorporates features of organisational modelling [Mue02], such as techniques for assignment of activities to resources in organisational structures.

To capture the dynamic nature of state transitions, a finite state automaton will be used to describe distinct cases within the system at some instant of time [Cri97] (see Section 2). At the semantic level, graphs will be used to represent system states and stochastic graph transformation will model state changes with non-deterministic timing, such as the execution of a business action with a known average delay or the assignment of an actor to a role (R1). This allows us to model semi-structured processes, where actions are not chosen using fixed control flow, but non-deterministically, influenced by deadlines and escalation events (R2-3). At the same time the visual, rule-based approach provides an intuitive notation for structural changes, which will be presented in Section 4. The states of the automaton will be attribute values used in pre- and postconditions of the transformation rules.

The operational semantics of stochastic graph transformation (GT) allow for simulation of workflows, including analysis of system architectures. The stochastic simulation [THR10] provides analysis capabilities for service-level metrics, such as how far past the deadlines late *cases* are and how effective is the use of load balancing and/or escalation handling.

Overall the methodology we are following consists of the following six steps. (1) Provide description of the case study, business environment and performance questions to be analysed; (2) Model the business process; (3) Define tests for the questions based on the model; (4) Assign probability distributions in order to be able to simulate; (5) Perform the simulations; (6) Analysis the results of the simulation.

Section 2 will discuss a pharmacy case study and will formulate questions to be answered by the simulation, addressing timeliness of the dispensing service and the efficiency of task allocation. We will then define our metamodel and illustrate the usefulness of the language by modelling the pharmacy business process using our graphical syntax in Section 4. In Section 5 we will show how to code or model in the textual syntax of the simulation tool, run simulations, and interpret the results.

2 Case Study

In order to validate and illustrate the requirements defined in Section 1, a pharmacy case study will be considered. Positions in the dispensary include registered pharmacists, pharmacy students, pharmacy technicians and cashiers. Formally, a person's position in the hierarchy of actors reflects their allowable role assignments (R1). In a chain of pharmacy stores, the work load varies across the different stores because it is dependent on the number of prescription requests at particular locations. The customer has a choice to pick up the filled prescription from the store or have it delivered to their home. If the pharmacy is extremely busy and the requested prescription has been ordered for delivery, then it can be transferred and filled by any pharmacy that is geographically nearby. This is an example of load balancing (R3). However, prescriptions ordered to be 'picked up' are restricted to be filled at the store location that the order was placed at.

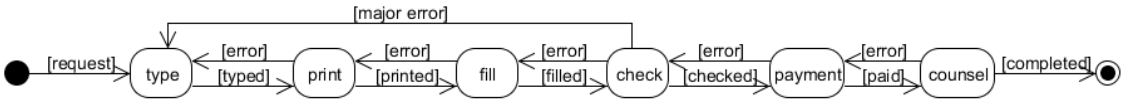


Figure 1: Finite State Automaton of the Pharmacy BP

The finite state automaton of a pharmacy business process is shown in Figure 1. Each state represents the stage of that particular case in the process. Pharmaceutical processes are safety critical and must be checked for correctness and accuracy. Hence, each task includes error checking to ensure that it is performed correctly (R4). For instance, if a state is skipped the process will backtrack to the previous state (R4). A typical process is initiated by a patient requesting a prescription to be filled. The patient is informed of an expected finishing time, resulting in a deadline for the case. The prescription is typed into the pharmacy database by an entry technician and the corresponding bottle label is printed. The prescription is filled by a filling technician and

checked by the dispensing pharmacist. The pharmacy cashier receives payment from the patient and then the patient is counselled by the dispensing pharmacist and given the filled prescription.

Temporal exception handling can be triggered when the case is approaching a deadline, reached the deadline, or exceeded the deadline. Escalations may also result in people overstepping their permissions and being temporarily assigned to a role required to deal with the exception (R2). An example is a prescription (case) which is 5 minutes past that deadline and has an escalation defined that permits a pharmacy cashier to temporarily take on the role as filling technician (R2).

The pharmacy case study will be used to illustrate concepts throughout this paper. Besides modelling the process we will be interested in answering questions about the timeliness of the service and the efficiency of task allocation, such as *Does escalation and/or load balancing increase the percentage of cases that are completed within a given deadline, or reduce the time that cases run past their deadline?* With these questions in mind we will explore different system features used in 4 different versions of the model as shown in Table 1. The results of these questions help to determine the most efficient variant of our model with respect to the chosen performance measures and will be further discussed in Section 5.

Escalation	**Load Balancing**	
	Yes	*No*
Yes	Version 1	Version 2
No	Version 3	Version 4

Table 1: Comparison of 4 Versions

3 Related Work

We will assess how far existing languages, workflow management systems, and simulation approaches satisfy the requirements (R1-4).

Business Process Modelling Notation (BPMN) [Gro09] and Web Service Human Task (WS-HumanTask) [ABO+07] both provide representations for role allocation, with BPMN using a graphical and WS-Humantask a textual syntax. BPMN also has a visual representation for escalation (R2), business roles (R1) and backtracking (R4) through *exception flows*, *swimlanes*, and *compensation flows* respectively. The exception flow denotes a flow that occurs outside normal flow based on a specific trigger (i.e. deadline exceeded) that redirects the flow. Swimlanes provide a static partitioning of activities that can be used to denote a fixed role allocation. Compensation associations are triggered if the outcome of an action is determined to be undesirable then it is necessary to 'undo' the activity.

WS-HumanTask can specify task *assignment* (R1), recoverable errors (R4), and *timeouts*, and triggers appropriate *escalation* actions (R2). Escalation handling is used in a manner comparable to our approach—however we use graph transformation rules to specify triggers and escalation levels specific to each case. Our language extends BPMN and WS-HumanTask by the use of stochastic durations, means to define organisational structures, and facilities to change the role assignment at runtime.

MILANO [AD00] is an example of a workflow management system providing a net-based modelling language to captures activities, roles, control flow and exception handling (R1-2) in a static and dynamic way. There are other flexible workflow management systems that support human collaboration, such as IBM's FlowMark [IBM98] and Xerox XSoft InConcert [Sar96]. FlowMark uses scheduling information of actors and roles at runtime. InConcert is based on

a graphical coordination model (R1,3). They both focus on role allocation in distributed systems using a control flow approach as opposed to a rule based approach, which will be further discussed in Section 4.

There are various simulation tools used in industry as well as a number of research-oriented environments, mostly based on the flow-oriented style of modelling. Little-Jil [Gro06] is a domain-independent visual agent coordination language for modelling and simulating processes. It captures key aspects of exception handling and deadlines (R2-3); however it's focus is primarily on the process steps, with very little emphasis on human resource allocation. ADONIS [BG10] is a simulation tool that captures general aspects of resource allocation (R1).

In summary, none of the approaches satisfy the requirements laid out in Section 1, with a majority missing backtracking (R4) and escalation handling (R2) as well stochastic modelling and analysis.

4 Rule-based Process Specification

In this section, our approach to rule-based process specification will be presented and illustrated by an application scenario. Some background information on graph transformation will be provided, but the reader is referred to [Hec06] for a more comprehensive introduction.

4.1 Metamodel and Notation

The linguistic [GH08] metamodel shown in Figure 2 captures the structure of the language. The notation is illustrated using the pharmacy business domain described in Section 2.

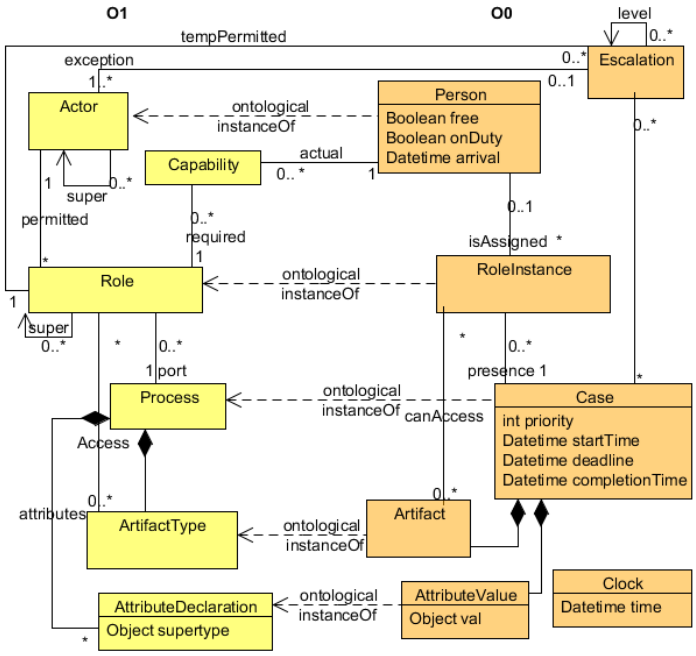


Figure 2: Metamodel (M2)

The metamodel combines type and instance-level concepts, located on opposite sides of the diagram for clarity, and related by (ontological [GH08]) "instance-of" relations. Thus, instances of this metamodel (M2) can represent both class-level (O1) and object-level (O0) features. The metamodel includes concepts of role-based access control, providing a representation of the relationship between actors and tasks of a business process [San98]. It was also influenced by a metmodel for business process models [AKR05], in particular with respect to organisational and task structure and resources. Similar to O1 and O0 in our metamodel, the reference also distinguishes two distinct categories of static and dynamic features. The static side is the model itself, while the dynamic side is the instantiation of the model.

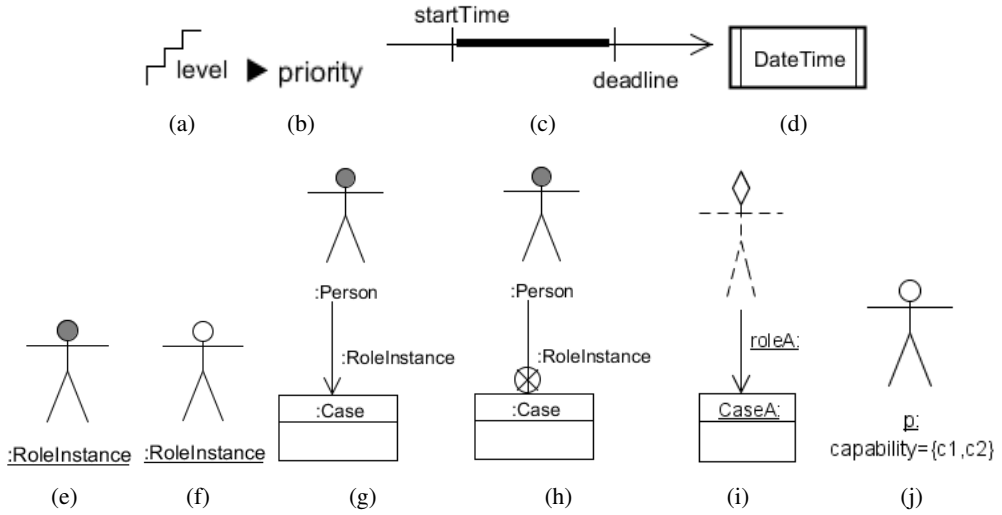


Figure 3: Notation for process configurations

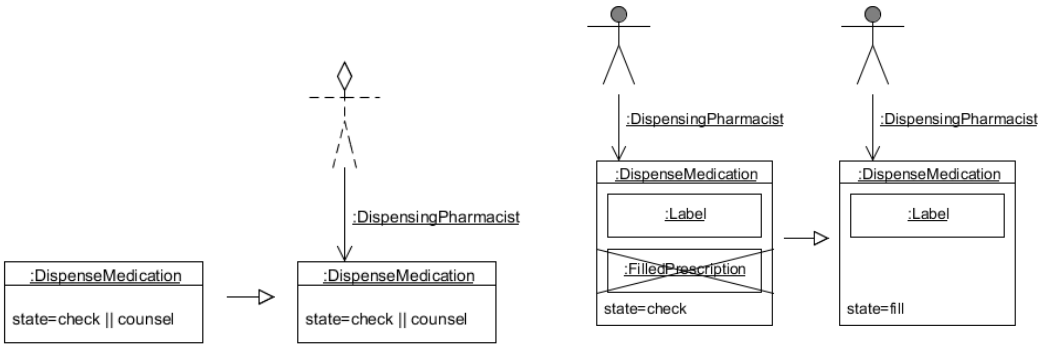
The notation for process configurations is pictorial and represents (constraints on) escalation levels, priorities, time and deadlines in states (instance graph) or state patterns (graph patterns to be matched by instance graphs). The syntax is introduced in Figure 3 showing (a) a Case's escalation level, (b) a priority attribute, and (c) the startTime and deadline attributes. The values corresponding to (a)-(c) are represented with a positive integer. If the values equal zero, then their corresponding icons will not appear in the Case. The current time in a state is shown as in (d). Subfigures (e)-(f) show a short-hand notation for the Boolean attribute free in metaclass Person: (e) for Person.free=false and (f) for Person.free=true. The representation of access rights is represented in (g) for internal and (h) for external access. If a person's access rights are internal, then they can access some of the artifacts contained in the corresponding case. The request for an assignment to be made is represented visually in (i). A Person's actual Capability is represented visually in (j). The syntax will be further illustrated in Section 4.2 and 4.3.

4.2 Graph Transformation Rules

Graph Transformation (GT) [Hec06] is a rule-based approach, which represents procedural knowledge in terms of a set of transformation rules defining preconditions and effects of the basic activities. This form of modelling is in contrast to the static *control-flow* approach, which focusses on the expected ordering of events. The Rule-based models are more suitable for flex-

ible processes because of the inherent non-determinism in selecting a rule and the objects it should be applied to purely based on the presence of a certain pattern in the configuration.

The transformation rules refer to the states of the finite automaton in Figure 1 as high-level control states. In particular, each prescription case contains a label corresponding to its current state in the process, determining the tasks that still need to be performed. State labels are also used to provide an external view of the case without immediately checking internal contents. However, this assumption can sometimes result in errors if the case is labelled incorrectly. Therefore, validation is required and sometimes results in backtracking transitions, as shown in the rule in Figure 4(b).



(a) Request Pharmacist (b) Backtrack check state

Figure 4: Graph Transformation Rules

The rules are a combination of domain specific actions and managerial actions. Domain specific action rules capture domain knowledge about the actions possible in the process and their preconditions and effects at the business-level. Examples are *type prescription*, *print label*, *receive payment* and *fill prescription*. Managerial rules are used to maintain the organisational structure in the business process. They include *role assignment*, *role request*, *backtrack* and *clock tick*.

For instance, the *fill prescription* rule states that if a prescription is not filled and there is a filling technician assigned to the case instance, then the fill prescription action can be performed. Since people do not always perform actions, *skip* rules are used to represent when an employee skipped a task. For example, an employee might decide not to fill a prescription and to pass it on to the next task, while still changing the state label. With the assumption that the state label is correct a pharmacist is requested to be assigned to the case without checking the internal contents, as shown in Figure 4(a). After the pharmacist is assigned, then the content is verified through pattern matching. If the case is missing an element, it is backtracked to the previous state as shown in the *check state backtrack* rule in Figure 4(b).

The model's local clock is incremented with a *clock tick* rule used to provide a model of time passing as required, for example, for checking deadlines. Syntactically, rules can access the current time through the time attribute in the Clock node to compute deadlines or validate conditions to trigger an escalation; hence if a case deadline has passed, the escalation level of the case should be increased.

4.3 Application Scenario

In the following we illustrate the graphical syntax of the language by presenting a sequence of instance graphs resulting from the application of rules defined in Section 4.2.

The scenario starts with the instance graph in Figure 5 on Monday 1 March 2010 at 11:34, with one escalated case in the *check* state, and is one minute away from its deadline. The available workers are a registered pharmacist and a cashier. A minute later (Figure 6) a temporal escalation is triggered on the case, raising the escalation level to 2 and temporarily permitting the cashier to take on the role as a filling technician for that case. The registered pharmacist is assigned to check the filled prescription. It is incomplete, resulting in backtracking to the *fill* state. The pharmacist leaves the case to type a new high priority delivery prescription and the cashier is assigned as a filling technician on the first case.

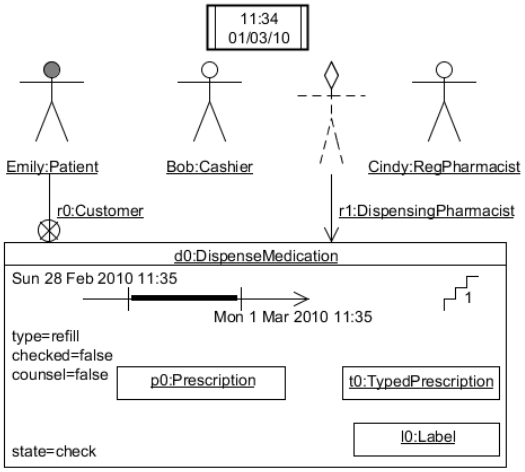


Figure 5: Monday 1 March 2010 at 11:34

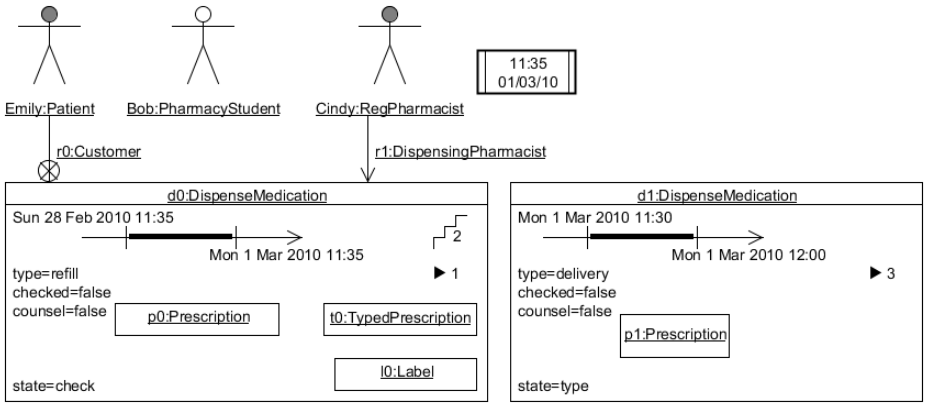


Figure 6: Monday 1 March 2010 at 11:35

5 Stochastic Graph Transformation Systems

Our models can be simulated using Graph-based Stochastic Simulation GraSS) [THR10], an extension of the Viatra [VIA] Eclipse-based model transformation tool. Our metamodel (Figure 2), model (type and instance graphs) and graph transformation rules are defined in Viatra2's programming language VTCL. Below is an example of the VTCL code for the *check state backtrack* rule as defined graphically in Figure 4(b).

```
gtrule BacktrackRule_checkState()= {
     precondition pattern lhs(Case_,State_) = {
      Case(Case_);
      AttributeValue(AttributeValue_);
            find RequiresChecked(Case_,AttributeValue_);
          find DPassigned (Case_,RoleInstance_,Role_,Person_);

            neg find FilledPrescriptionExist(Case_,Artifact_,ArtifactType_);
            Case.state(State_);
            Case.attr4(R1,Case_,State_);
            check (value(State_)== "check");
           }
           action {
          setValue(State_,"fill");
            println("error (backtrack to fill state)");
           }
   }
```

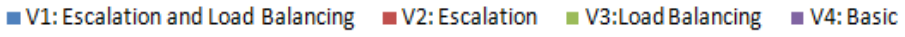
The tool executes models derived from the rules in Section 4.2 with corresponding stochastic timing added. Results are determined by *probes*, i.e., patterns matching situations such as incomplete cases in each intermediate graph, as well as by counting the total number of applications of particular rules during a simulation run. The average times for activities in the pharmacy process were defined according to [SL05, Tru06]. Rules for creating new cases have been given exponential distributions while the remaining rules were defined using normal distributions. The results represent 2.77 hours of simulated time based on the number of times the *clock tick* rule was applied (166 mins). The number of steps was limited to 2500, with a batch size of 3, and was run on four distinct versions as described in Section 2.

Two of the versions implemented escalation handling with three predefined levels. The escalation would be triggered if the case was within 5 mins to deadline, at the deadline or more than 5 mins past the deadline. At escalation level 1, pharmacy cashiers are temporarily permitted to be assigned as an entry technicians for that particular case. At escalation level 2, pharmacy cashiers are temporarily permitted to be assigned as filling technicians. At escalation level 3 untrained pharmacy students are temporarily permitted to be assigned as filling technicians and/or entry technicians. The other two versions implement load balancing across two pharmacies, providing the option to transfer prescriptions to the another store.

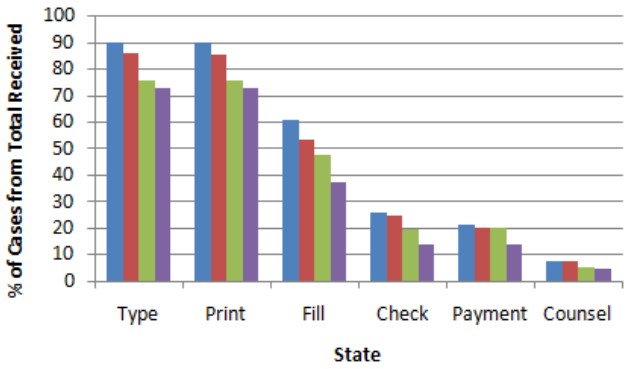
All of the four versions were tested on the same model (instance graph). The initial model was composed of two pharmacies. The first pharmacy consists of one registered pharmacist, two technicians, one cashier, one pharmacy student and two active dispense medication cases. The second pharmacy consists of two registered pharmacists and one technician.

The bar graphs below visually represents the comparison of results obtained for the four versions. The questions raised in Section 2 are answered as follows. The percentage of cases that

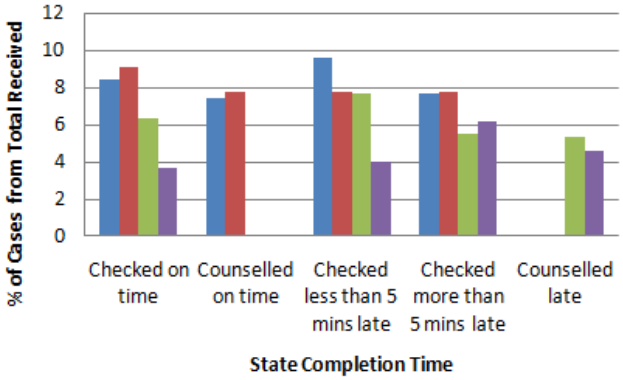
completed states in a process within 2.77 hours is presented in Figure 7(b). The percentage of cases that completed the check and counsel states on time, versus the range of lateness for the completeness of check and counsel states, is presented in Figure 7(c). The results show that the addition of escalation and load balancing has a positive influence on the probability of completing on time. This confirms the expected tendency, i.e., that version 1 (escalation and load balancing) results in the most favourable performance.



(a) Legend



(b) State Completion Comparison



(c) Completion Time Comparison

Figure 7: Results

6 Conclusion and Future Work

In this paper, we introduced a visual modelling language which allows us to represent humans as part of flexible workflows using a rule-based approach. The language is based on a notion of global time and stochastic delays for rule applications. We explored the stochastic behaviour of

the model through simulation [THR10] to take performance measures such as the probability for cases to finish within their deadlines, comparing systems that implement load balancing and/or escalation handling.

Our results illustrated that a combination of escalation handling and load balancing would improve the business process. These quantitative findings suggest various improvements that can take place in a busy pharmacy, such as increasing staff, transferring prescriptions to the owner's other nearby pharmacy, and temporarily permitting people to take on higher roles as far as permitted by law.

In the future, we plan to continue to explore the stochastic behaviour of the model by comparing the use of assignment polices and different scheduling protocols (i.e. by priority or deadline). Scalability will be evaluated through simulations on larger models.

Acknowledgements: The authors would like to thank Georgina Donyina, Pharmacist/Owner of a pharmacy located in Canada, for going through the pharmacy business process.

Bibliography

[ABO+07] Adobe, BEA, Oracle, A. Endpoints, IBM, SAP. Web Service Human Task (WS-HumanTask). Technical report 1.0, June 2007.

[AD00] A. Agostini, G. De Michelis. Improving Flexibility of workflow Management Systems. In Aalst et al. (eds.), *Business Process Management*. Lecture Notes in Computer Science 1806, pp. 289–342. Springer Berlin Heidelberg, 2000.

[AKR05] B. Axenath, E. Kindler, V. Rubin. The Aspects of Business Processes: An Open and Formalism Independent Ontology. Technical report tr-ri-05-256, Computer Science Department, University of Paderborn, 2005.

[BG10] BOC-Group. ADONIS:Community Edition. *URL: http://www.adonis-community.com/*, 2010.

[Cri97] J. M. Crichlow. *An Introduction to Distributed and Parallel Computing*. Prentice Hall, 1997.

[Don10] A. Donyina. Stochastic Modelling and Simulation of Dynamic Resource Allocation. In Ehrig et al. (eds.), *ICGT*. Lecture Notes in Computer Science 6372, pp. 388–390. Springer, 2010.

[GH08] C. Gonazalez-Perez, B. Henderson-Sellers. *Metamodelling for Software Engineering*. John Wiley & Sons Ltd., 2008.

[Gro06] L. P. W. Group. Little-JIL 1.5 Language Report. Technical report, Laboratory for Advanced Software Engineering Research, University of Massachusetts, Amherst, 1997-2006.

[Gro09] O. M. Group(OMG). Business Process Modeling Notation (BPMN). Technical report Version 1.2, January 2009.

[Hec06] R. Heckel. Graph Transformation in a Nutshell. In *Electr. Notes Theor. Comput. Sci.* Volume 148(1), pp. 187–198. 2006.

[IBM98] IBM. *Image and Workflow Library FlowMark Design Guidelines*. Volume 2.3. IBM Redbooks, 1998.

[Mue02] M. z. Muehlen. *Workflow-based Process Controlling: Foundation,Design, and Application of Workflow-driven Process Information Systems*. Logos Verlag Berlin, 2002.

[San98] R. S. Sandhu. Role-Based Access Control. In *Adv. in Computers*. Volume 46, pp. 237–286. Academic Press, 1998.

[Sar96] S. K. Sarin. Workflow and Data Management in InConcert. *Data Engineering, International Conference on* 0:497, 1996.

[SL05] C. W. Spry, M. A. Lawley. Evaluating hospital pharmacy staffing and work scheduling using simulation. In *WSC '05: Proceedings of the 37th conference on Winter simulation*. Pp. 2256–2263. Winter Simulation Conference, 2005.

[THR10] P. Torrini, R. Heckel, I. Ráth. Stochastic Simulation of Graph Transformation Systems. In *Fundamental Approaches to Software Engineering (FASE)*. Lecture Notes in Computer Science 6013, pp. 154–157. Springer Berlin / Heidelberg, 2010.

[Tru06] I. Truter. Dispensing Service Research-Pilot Project. *Pharmaciae- Official publication of the South African Pharmacy Council* 14(1):20–23, April 2006.

[VIA] VIATRA2. VIsual Automated model TRAnsformations framework. *URL: http://wiki.eclipse.org/VIATRA2*.

[Ze96] A. Y. H. Zomaya, editor. *Parallel & Distributed Computing Handbook*. McGraw Hill, 1996.