Electronic Communications of the EASST
Volume 38 (2010)

Proceedings of the Fifth International Conference on Graph Transformation - Doctoral Symposium (ICGT-DS 2010)

Modelling Flexible Human Resource Allocation by Stochastic Graph Transformation

Adwoa Donyina and Reiko Heckel

17 pages

Guest Editor: Andrea Corradini
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Modelling Flexible Human Resource Allocation by Stochastic Graph Transformation

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:

In order for business process models to accurately capture the flexibility of human actors, we have to allow for flexible task management and role allocation. That means to take into account deadlines and escalation procedures, consider access rights, but allow temporary promotion and dynamic assignment of actors. In order to meet these requirements we will represent the configurations of a business process as graphs and operations changing these configurations by graph transformation rules. We will propose a domain-specific notation to visualise rules and configurations.

To analyse performance and reliability properties of such models, we use stochastic simulation. The approach is illustrated by means of a pharmacy case study.

Keywords: stochastic modelling; graph transformation; organisational metamodelling; business processes; escalation handling; non-deterministic time; dynamic role (re)-assignment; distributed systems; load balancing; role based access control; process scheduling; assignment policies

1 Introduction

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 overrule procedure. When modelling such processes, we have to deal with a variety of other issues such as access control, resource (re)assignment policies, promotion to and demotion from roles, deadlines, priorities and escalation handling.

Most existing methods and tools are not addressing the complexity of human behaviour in business processes, because the formal approaches that have been developed so far in software engineering focus on technical components of a business process. This is achieved by modelling how software and technical components will react to triggers issued by a software component and perform designated actions.

Businesses need a means to test scheduling protocols, policies and regulations, prior to exposing it to the current methods used in their business. With the goal of increasing business

productivity, simulations will provide analysis facilities for gathering service level guarantee metrics and policy comparisons. It will be helpful if these standardized methods were visualized for training facilities.

In this paper we will be describing our approach to modelling flexible human resource allocation by stochastic graph transformation. We will first define our requirements then formulate a feature diagram (Figure 1). We will then provide an overview of the ingredients of our approach.

Resource allocation principles are based on the objective of completing a work item as quickly as possible, through order of case selection and selection of resource to perform the case. Therefore, the language feature requirements have been divided into two groups: features that help selection of tasks and features that help selection of people for the task (role assignment).

1. **TASK MANAGEMENT**

   R1.1 *Process scheduling influenced by deadlines and priorities*: Scheduling should maximise throughput and minimise the number of tasks completed past the deadline.

   R1.2 *Non-deterministic duration of tasks*: The time it takes to complete a task varies depending on factors outside the scope of the model. We will therefore require a stochastic approach. 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.

   R1.3 *Temporal escalation handling*: Deadlines can trigger escalations, which lead to resources being diverted to the escalated case. Procedures should be in place as to handle escalations so that they comply with legal requirements and are allowed to react efficiently.

   R1.4 *Load Balancing*: Tasks can be dynamically (re)assigned globally to other sites (locations), this should maximise throughput and minimise the number of tasks completed past the deadline.

2. **ROLE ALLOCATION**

   R2.1 *Dynamic (re)-assignment*: Within distributed systems, tasks can be dynamically (re)assigned locally to people based on their availability and capabilities. Optionally, assignment policies can be defined. In particular, predefined assignment protocols can be used to determine who should be assigned to a particular role, such as the least-qualified person able to play it.

   R2.2 *Role promotion, demotion and temporary promotion*: A promotion occurs once a person acquires capabilities sufficient to be assigned a new role. Demotion means that a person loses such permission. Temporary promotion occurs in response to an escalation for a limited period of time.

   R2.3 *Access control*: Security protocols that distinguish people's permissions to perform roles and associates roles to their corresponding access rights to artifacts such as documents or files.

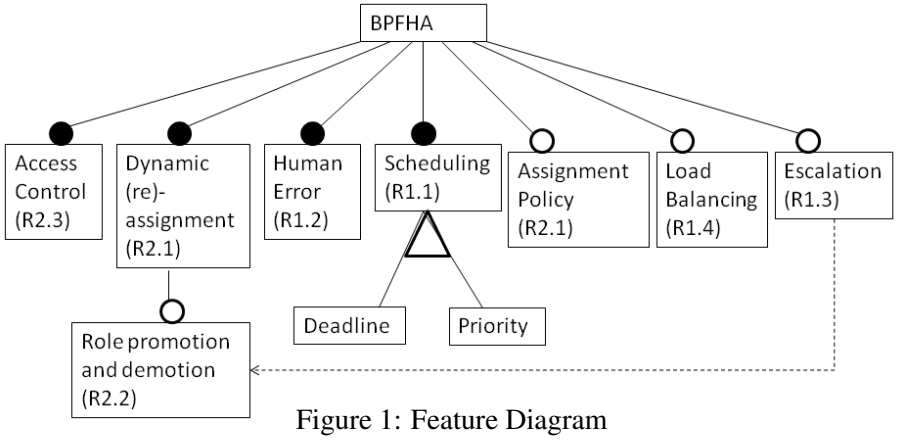


Figure 1: Feature Diagram

The requirements are summarized in the feature model [KOD10, ML05] defined in Figure 1. The mandatory features, such as access control and human error are denoted with a shaded black dot. The optional features, such as assignment policy and escalation are denoted with an unshaded dot. Sub-features deadline and priority have an alternative (xor) relationship under the mandatory scheduling feature. The dependency implication is denoted with the dashed arrow, which implies that if the source feature (i.e escalation feature) is selected then the target feature (i.e role promotion/demotion feature) must be selected as well.

To manage the complexity of such 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. We designed a graphical language for modelling human actors in business processes using a rule-based approach. The language syntax of the domain specific language (DSL) is defined using a metamodel. The metamodel is influenced by Role-Based Access Control (RBAC) [San98] models while the concrete notation extends that of UML class and use case diagrams (R2.3). The domain-specific language incorporates features of organisational modelling [Mue02], such as techniques for assignment of activities to resources in organisational structures (R2.1,2.2).

To capture the dynamic nature of local state transitions of cases, 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 are 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 (R1.2,1.4,2.1). This allows us to model semi-structured processes, where actions are not chosen using fixed control flow, but non-deterministically, influenced by deadlines, priorities and escalation events (R1.1,1.3,1.4). At the same time the visual, rule-based approach provides an intuitive notation for structural changes, distinguishing between domain-specific and domain-independent rules. This enables the specification of generic allocation policies, defined using application conditions and constraints [HW95] (R2.3), which will be presented in Section 5. The states of the

automaton will be attribute values used in pre- and postconditions of the transformation rules. The transformation rules will capture the dynamic configuration of the system. The modelling approach being proposed is called Business Process of Flexible Human Actors (BPFHA).

The operational semantics of stochastic graph transformation (GT) allow for simulation of workflows. The stochastic simulation [THR10] provides analysis capabilities for service-level metrics, such as how far past the deadlines late *cases* are or the probability to complete a task within a deadline. It also enables us to compare the performance of different policies and the effectiveness of additional features such as load balancing and/or escalation handling (R1.1,1.2,1.4).

We will discuss how existing methods (fail to) satisfy the requirements in Section 3. Overall the methodology we are following is composed of a four stages process. (1) Business Processing Modelling, (2) Process Design, (3) Process Encoding, and (4) Performance Evaluation Stage. This methodology will be described in detail in Section 4 and illustrated with the use of a pharmacy case study in Section 2. The process design and performance evaluation stages with be illustrated in Section 5 and Section 6.

2 Case Study

In order to validate the requirements defined in Section 1 and illustrate the methodology, a pharmacy case study will be considered. The people involved have different levels of access rights depending on their qualification or role (R2.3). This process is about dispensing medication at a pharmacy, which occurs at a dispensary. Positions in the dispensary include registered pharmacists, pharmacy students, pharmacy technicians and cashiers. The dispensary also requires external involvement of customers, in the role of patients with basic access rights (R2.3). A person's position in the hierarchy of actors reflects their allowable role assignments (R2.1,2.2). For instance if a filling technician role is required to be assigned and both a registered pharmacist and a pharmacy technician are available, the technician should take on that role (R2.1). However in a situation where no pharmacy technicians are available, the pharmacist is flexible enough to take on the filling technician role (R2.1).

Workers are likely to react to emergencies by stopping their predetermined routine schedule of work (R1.3). An example is an urgent prescription that needs to be dispensed by a pharmacist while the current pharmacist on duty is busy. In such a case, the urgent prescription is given priority over other tasks (R1.3). Escalations may also result in people overstepping their permissions and being temporally assigned to a role required to deal with the exception (R1.3,2.2).

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 (R1.4). However, there is a restriction that prescriptions ordered to be 'picked up' at a store can only be filled at the store.

The finite state automaton of a pharmacy business process consists of the following sequence of states: *type*, *print*, *fill*, *check*, *payment* and *counsel*. 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. For instance, if a state is skipped, the process will backtrack to the previous state (R1.2). 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 earliest due date (EDD) and priority based scheduling protocols assist in determining the order prescriptions get filled (R1.1). 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. The deadlines are approximations based on the method of placing an order and the degree of the customer's urgency (R1.1). For instance, if a customer decides to wait in the store, then the prescription is expected to be finished within 15 mins. If the customer decides to return to pick up the prescription, then the prescription is expected to be finished within 1hr. If the prescription was requested online, then it is expected to be completed within 24 hrs, etc. On the other hand, if the prescription is for home delivery, then it must be finished at set times for the delivery service to pick it up.

Temporal exception handling can be triggered when the case is approaching a deadline, has reached the deadline, or has 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 (R1.3). 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 (R1.3).

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. The list of questions that we would like to consider is as follows:

1. Which scheduling method (by priority/deadline) increases the probability for a prescription to be completed on time? How does it affect the number of cases completed out of the total received?

2. Does the usage of assignment policies decrease the number of idle workers (not assigned to jobs) at any point in time?

With these questions in mind we will explore different system features defined in the feature model as shown in Figure 1. The results of these questions help to determine the most efficient variant of our model with respect to the chosen performance measures.

3 Related Work

We will assess how far existing languages, workflow management systems, and simulation approaches satisfy the requirements.

Business Process Modelling Notation (BPMN) [(OM09] 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 (R1.3), business roles (R2.1) and backtracking (R1.2) 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* (R2.1), recoverable errors (R1.2), and *timeouts* (R1.2), and triggers appropriate *escalation* actions (R1.3). 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 capture activities, roles, control flow and exception handling (R1.3,2.1) 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 (R1.1,2.1). 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 5.

There is 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 (R1.1,1.3); 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 (R2.1).

In summary, none of the approaches completely satisfy the requirements laid out in Section 1 with a majority missing dynamic (re)-assignment and load balancing (R1.4, 2.1), as well as stochastic modelling and analysis.

4 Methodology

This section will discuss the methodology needed by developers to employ the project's new modelling approach which uses simulations to test scheduling protocols, policies and regulations.

In order to use the modelling language at its full potential it is ideal that the workflow that has been already selected consists of human involvement with the additional properties of role hierarchies, role allocations and access rights. The workflow in mind should be written in case study format, clearly identify various actors (job positions) and roles that can be assumed by them, with specifications of users' access rights to artifacts. This written documentation should also describe existing and potential future policies, protocols, and procedures. The chosen protocols, policies and regulations should range from the current available model features of: assignment policies, load balancing, escalation handling and scheduling protocols.

With the assumption that a workflow problem has been selected and written as explained in Section 2, the following four consecutive methodology stages need to be performed consisting of: business modelling, process designing, process encoding and performance evaluation stage. The business modelling stage consists of defining business domain in terms of UML class and use case diagrams; creating an artifact centric view using finite state automata; and creating a process centric view using activity diagrams. The process design stage consists of: extending the UML class and use case diagrams; defining instance graphs and domain specific GT rules in BPFHA concrete syntax; selecting predefined managerial GT rules in language concrete syntax; selecting distributions for each rule. The process encoding stage involves: defining metamodel and model in Viatra [VIA]; selecting predefined mangerial GT rules in Viatra textual syntax; translate probe and domain specific GT rules in Viatra textual syntax; encoding simulation parameters. The final performance evaluation stage consists of: performing stochastic simulations in order to analyse the results and formulate conclusions.

5 Process Design

The language is based on the rule-based approach of graph transformations [EPS73] which represents knowledge in terms of a set of "IF-THEN" *rules*. We use graphs for the foundation of our graphical domain specific language because of the benefits of using a rule based approach in addition to a control flow approach. The rule-based form of modelling is in contrast to the static *control flow* approach, which represents precision in the expected ordering of events. The control flow approach is suitable for defining static aspects in the business level; however it is not suitable for low level dynamic managerial aspects such as role assignment and scheduling, because in order to capture these non-deterministic decisions it would have to break down into various complex exceptions. Hence, our language uses rule-based approach in order to capture the non-deterministic complexity of the management level, yet encodes control flow in the graphs to capture the business level aspects. Hence, the language contains an aspect of control to define predefined routines, yet is flexible enough to represent non-deterministic managerial decisions by allowing humans to select rules to apply for process design experts to use.

The BPFHA domain specific language is used to model concepts from a pharmacy organisational structure problem domain. Section 5.1 will model role hierarchies and artifact concepts in terms of extended UML class and use case diagrams. Section 5.2 will define transformation rules, whereas Section 5.3 will discuss distribution selection for each of the rules.

5.1 Artifacts and Role Hierarchy

The role hierarchies and artifact concepts are defined in a type graph. The pharmacy type graph is a combination of Figures 2 and 3, illustrating the actor hierarchy, role association, required capabilities and temporarily permitted roles. The concrete syntax extends the UML use case and class notation. UML class diagrams describe types of objects in a system and their static relationship, whereas UML use case diagrams present a graphical overview of the functionality provided by a system in terms of actors, their goals. The goals are defined as use cases which are denoted as ovals within the use case diagram and described using use case descriptions, as

illustrated in Table 1. The actor generalization is denoted with an arrow ending in a hollow triangle, which is used to define inheritance between actors. Figure 2 illustrates how to visualise capability and escalation constraints for actors to take on roles.

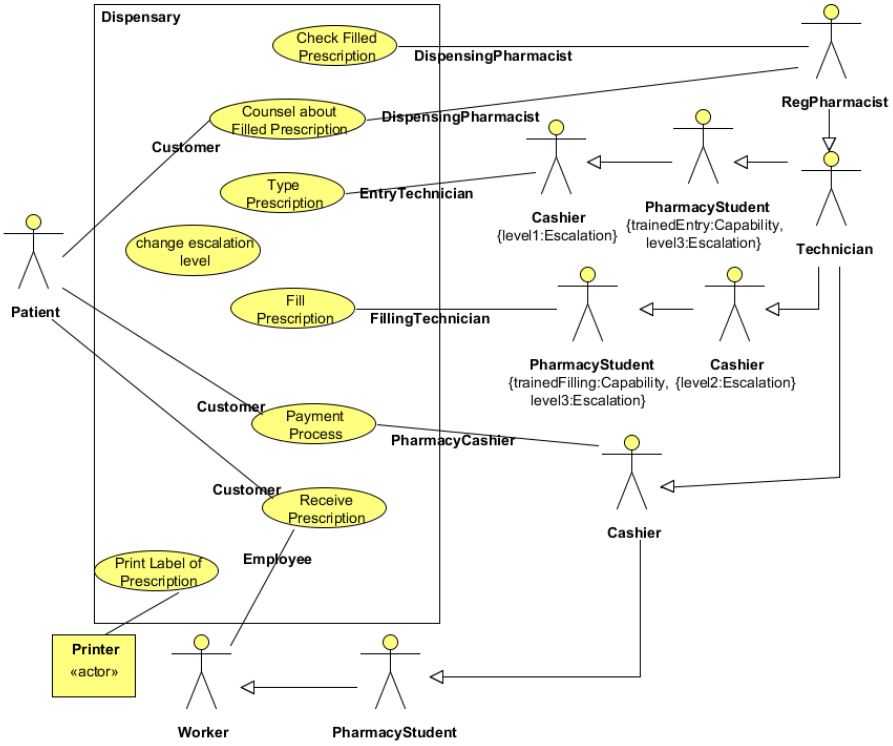


Figure 2: Role Hierarchy Model

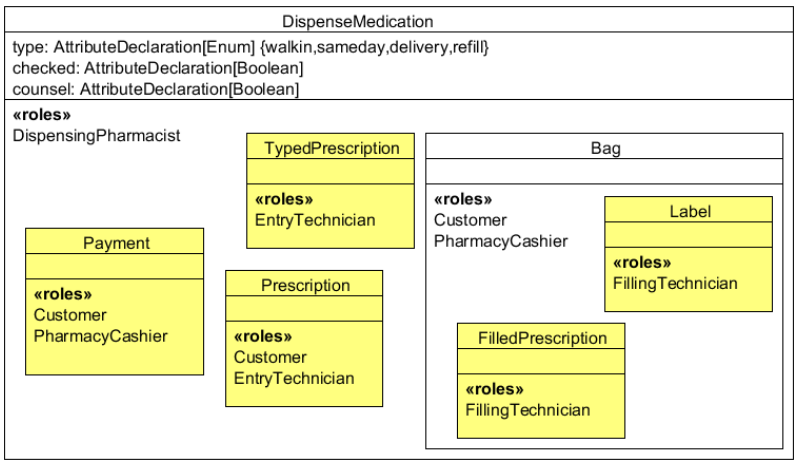


Figure 3: Artifact Model

The access control feature (R2.3) is defined in both of these models. In the role hierarchy model, UML's actor hierarchies and actor — use-case relations were used here, such that the roles are defined at the endpoints of the lines connected to the use case. The hierarchy model can also be used to define capability and escalation constraints; by defining set notation underneath the actor symbol, such that if at least one constraint holds true, then the actor can take on a given role.

In the artifact model, actors in the hierarchy model were referred to under the stereotyped keyword ≪role≫ used to define access rights of actors to artifacts. If an actor is assigned to a particular role for a particular *DispenseMedication* case instance, then the actor can access the corresponding artifacts. The artifact model defines Attribute declarations in the first section of the classes, such as 'counsel:Boolean'. Other attributes such as states, start time and completion time were defined in the metamodel level of this language.

5.2 Transformation Rules

The transformation rules are based on graph transformation rules, which will be separated into domain specific actions and managerial actions defined independently of the domain. Domain specific rules capture business level knowledge about the process, whereas managerial rules are used to maintain the organisational structure in the business process.

Name:	Fill Prescription
Primary Actor:	Technician
Goal (of the User):	successful print of prescription label
Precondition:	prescription label exists without filled prescription
Postcondition (successful execution):	prescription filled
Trigger Event:	label arrived at filling station
System (implementing it):	Dispensary
Participating Actor:	Technician

Table 1: Fill Prescription Case Details

A use case diagram and its corresponding use case descriptions can be used to realise the domain specific rules, such that each use case corresponds to at least one rule and the use case description pre and post conditions informally describe the contents of the rule. For instance, Table 1 presents a subset of the use case description for the fill prescription use case and Figure 4 defines the corresponding GT rule.

Figure 4 illustrates the *fill prescription* domain specific rule which denotes 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. Each prescription *case* contains a state label, which corresponds to the task that needs to be performed, with the assumption that upon completion of each task the state label should correctly represent the state transitions. Hence, the domain specific transformation rules use the states described in Section 2 as high-level control states. Other domain specific rules which correspond to a use case in the extended use case diagram defined in Figure 2 include: *type prescription*, *print label*, and *receive payment*.

Some of the managerial rules include *role assignment*, *role request* and *clock tick*. The *clock*

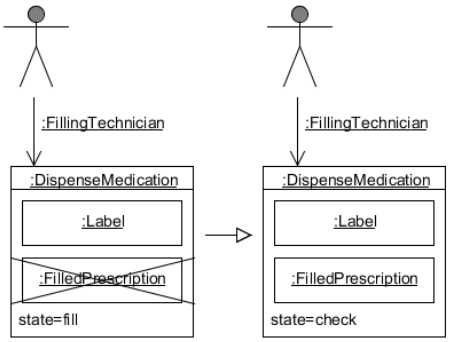
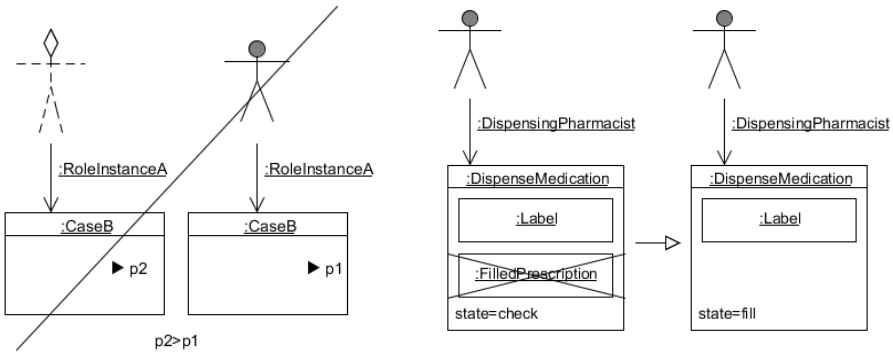


Figure 4: Fill Prescription Rule

tick rule is used to represent a model of time to capture aspects such as deadlines. A notion of continuous time was selected as a basis for stochastic delays of rule applications [HT10]. 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*. The local clock has a *clock tick* rule that increases the *DateTime* by minutes. If a *case* deadline is passed, the escalation level of the case should be increased. The state labels are also used in the managerial rules 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, error checking is required and sometimes results in backtracking transition to correct the case, thus *backtrack* rules are required.

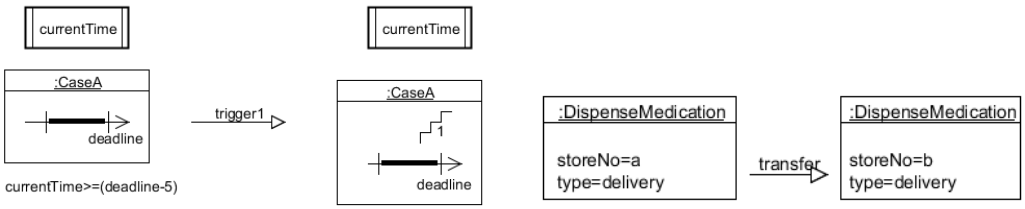
Below we will map the managerial rules to the corresponding language feature requirements. The scheduling feature (R1.1) provides the option to schedule according to deadlines or priorities. Figure 5(a) shows how to schedule by priority using global constraints [HW95]. Skip and backtrack rules are used to represent the human error feature (R1.2). The skip rule changes the case state label, without changing the case artifacts. Whereas, the backtrack rule checks that the state label is consistent with the case artifacts, as shown in Figure 5(b).



(a) By Priority (b) Backtrack check state

Figure 5: Task Management (R1.1-1.2)

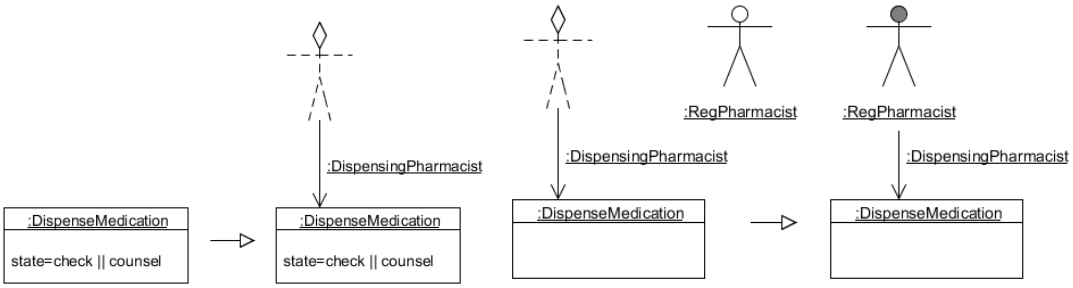
The escalation feature (R1.3) requires trigger and temp assignment transformation rules. The trigger rules are used when a particular case has gone into an escalated state, such as approaching a deadline (Figure 6(a)). The load balancing feature (R1.4) uses a transfer transformation rule to distribute a prescription to another store. The transfer rule only transfers a prescription if it is for delivery, as shown in Figure 6(b).



(a) Trigger to escalation level 1 (b) Load Balancing Rule

Figure 6: Task management (R1.3-1.4)

The dynamic (re)-assignment (R2.1) feature requires request, assignment and unassignment rules. The request rule uses local state labels to determine if an actor is required for a particular case (Figure 7(a)). The assignment rules use the post condition of request rules to assign actors to roles (Figure 7(b)). Similarly, the unassignment rules use the state label to determine if the actor has finished performing his or her duties. The assignment policy feature can be incorporated into assignment transformation rules. An assignment policy defines an additional condition to assignment rules, for instance if the assignment policy states that the least qualified available worker able to do the job should be assigned, then additional negative application conditions would be required. Temp assignment rules (R2.2) occur only at escalated states by temporarily permitting a person to take on a role that he or she are not normally allowed to perform.



(a) Request Pharmacist (b) Assign Pharmacist

Figure 7: Role Allocation R2.1

5.3 Stochastic Graph Transformation System

A stochastic graph transformation system [HEGH11] is a graph transformation system that associates general distributions to rule-match pairs using generalised semi-Markov processes. The general distributions include normal and exponential distributions.

This section will illustrate how to assign general distributions to the rule-match pairs (event) for the current graph transformation system, which was defined in Section 5. The distribution values can be visually denoted on the GT rule transition arrow.

The rule actions need to be categorized by types of distributions, which is chosen based on the time data that has been gathered on that particular action. For instance, if we know the rate of arrivals such as time it takes before receiving a new telephone call, then it can be defined as an exponential distribution. On the other hand if we know the average length of time it takes to complete an action, such as the average time to cut a person's hair then it can be categorized as a normal distribution or a lognormal distribution, which defines a positive normal distribution. Hence, each GT rule also needs to be categorized into exponential or lognormal distribution, and then associated with specified scheduled delays.

The exponential distribution requires a corresponding rate, whereas the lognormal distribution requires mean and variance values. The corresponding rate, mean and variance parameters for the case study were selected based on measured pharmacy process data from [SL05, Tru06, SGB81] and discussions with pharmacy experts.

Distribution data was gathered from a dispensing service pilot project [Tru06], which took place from 27 November 2005 to 4 December 2005. The sample size of the project consisted of 29 pharmacies. The dispensing procedure was separated into three phases. The first phase involved entry and evaluation (checking) of the prescription. The second phase involved filling, labelling and checking filled prescription. The third phase involved counseling the patient. Table 2 displays the average time spent per prescription in each phase, which resulted in the average total time of dispensing a prescription to be 16 minutes and 59 seconds with a standard deviation of 8 minutes and 29 seconds.

Phases	**Average time (min spent per prescription**	**Minimum (min)**	**Maximum (min)**
Phase 1	9 min 11 sec ($\pm$ 5 min 37 sec)	1 sec	34 min 12 sec
Phase 2	4 min 37 sec ($\pm$ 4 min 51 sec)	15 sec	40 min 54 sec
Phase 3	3 min 9 sec ($\pm$ 2 min 34 sec)	4 sec	14 min 10 sec
Total	16 min 59 sec ($\pm$ 8 min 29 sec)	3 min 23 sec	54 min 10 sec

Table 2: Average time spent per prescription [Tru06]

The descriptions of our pharmacy business process were mapped to the detailed dispensing procedure used in the pilot study. With the help from pharmacy professionals we concluded our estimates as shown in Table 3. Each state corresponds to an event, which is the pattern match of a defined GT Rules. Since the states are dependent on preceding actions to occur, the majority of the domain specific GT Rules will be categorized to a lognormal distribution; hence mean and variance is shown.

The estimated average times for the following exception events: skip, backtrack, escalation

State	Event	Mean μ (min)	Variance σ^2 (min)
Type	TypePrescription GT Rule	1.25	0.4375
Print	PrintPrescription GT Rule	0.26	0.024806
Fill	FillPrescription GT Rule	1.17	0.395833
Check	SuccessfulCheck and UnsuccessfulCheck GT Rule	2.00	0.8125
Payment	GivePayment GT Rule	0.71	0.047292
Counsel	Counsel GT Rule	1.17	0.583333

Table 3: Estimated average times to complete states

trigger and transfer (load balancing) were all influenced by the estimated average times to complete the states in the pharmacy business process (Table 3). For instance the mean of the skip events were calculated to be 0.25 factor higher than their corresponding fulfilled action distribution defined in Table 3 (μ_{skip}=$\mu_{fulfilled}$+0.25) i.e. the mean for the SkipTypePrescription event is μ=1.50. On the other hand, the means for the backtrack events were set to the same mean as the fulfilment of the state defined in Table 3 using the assumption that it will take the same amount of time to detect an error in a case as it would take to perform the action. Similarly, the optional load balancing and escalation handling features are partially influenced by the other determined estimations.

The arrival of new prescription cases was categorized as an exponential distribution, because the random arrival of prescription orders is independent of other actions. The estimation values were influenced by data collection from a medium-sized New Jersey Hospital pharmacy unit [SGB81]. Their results stated that the mean arrival rate for high priority orders was 0.096, whereas the mean arrival rate for regular orders was 0.259. After discussion with pharmacy professionals estimations shown in Table 4 were decided.

Event	Frequency	Mean λ (mins)	Mean arrival rate λ^{-1}
NewCase GT Rule	Every 5 mins	10	0.1
NewCaseMedPriority GT Rule	Every 10 mins	20	0.05
NewCaseHighPriority GT Rule	Every 30 mins	60	0.01667

Table 4: GT Rule (Exponential Distribution)

The events that correspond to managerial GT rules were categorized as lognormal distributions because an action such as assignment is dependent on the request for assignment action. Since the domain specific actions only represented a proportion of the total dispensing time specified by the pilot research data in Table 2, the remaining dispensing will be covered with the managerial actions.

6 Performance Evaluation

Graph transformation systems can be simulated in a tool supporting Graph-based Stochastic Simulation (GraSS) [THR10], an extension of the Viatra [VIA] Eclipse model transformation plugin. Our language is defined in Viatra2's VTCL programming language. The results are outputted into reports of the observed rule frequency and average probe occurrence. The questions raised in Section 2 can be answered by performing simulations on the 4 unique model versions outlined in Table 5. The models in versions 1-4 compare the use of assignment policies and different types of scheduling (Question 1-2).

Assignment	**Scheduling**	
	Deadline	*Priority*
No Policy	Version 1	Version 2
Policy	Version 3	Version 4

Table 5: Comparison of 4 Versions

After running the simulations on the 4 model versions, the statistical results need to be analyzed to formulate answers. The set of simulation results are gathered in batches with the number of probe pattern matches and number of executions of action rules. For example, if the probe counts number of idle workers ('PersonAvailable' probe), then in each intermediate simulation step the total number of idle workers will be counted. This data will be used to determine the average number of idle workers in a simulation. This probe rule can be used to answer question 2 by analyzing the smallest probe result in version 1-4.

Two predefined probe rules can be used to answer the first half of question 1, 'latefromtotal' and 'lateCases'. The 'lateFromTotal' probe calculates the numbers of cases that are late from the total number received, whereas 'lateCases' counts the number of late cases. In order to answer the second half of question 1, the batch rule reports will need to be analyzed in order to calculate the number of times the 'successfulCheck' rule was applied in relation to the sum of the execution of the new case rules. Therefore, the answer will be gathered from the comparison of version 1-4 (number of successful prescription checks divided by the total number of new cases received).

The average simulation runtime was 30 minutes representing 3 hours of simulated time based on the average number of times the *clock tick* rule was applied (198.4125 minutes). The number of simulation 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 cases were without specific assignment policies, with scheduling influenced by deadlines or priorities. The other two included the effect of the policy in addition to the choice of scheduling strategy. The initial model for each simulation run contained one registered pharmacist, three technicians, one cashier and one active dispense medication case.

The bar graphs below visually represent the comparison of results obtained for the four versions, whereas the probability for a prescription to be completed late is presented in Table 6. The questions raised in Section 2 are answered as follows. The average number of idle workers is presented in Figure 9. The results show that assignment policies and scheduling choices have an effect on idle workers while the influence on the probability of completing on time is limited. Therefore an additional measure was used to determine the

Version	***P*(Case is late)**
V1	0.247
V2	0.256
V3	0.245
V4	0.244

Table 6: *p*(prescription case is completed late)

number of completed cases at the end of the 3 hour period. The results are shown in Figure 8 and confirm the expected tendency, i.e., that version 4 (assignment policy and scheduling by priority) resulted in the most favourable results.

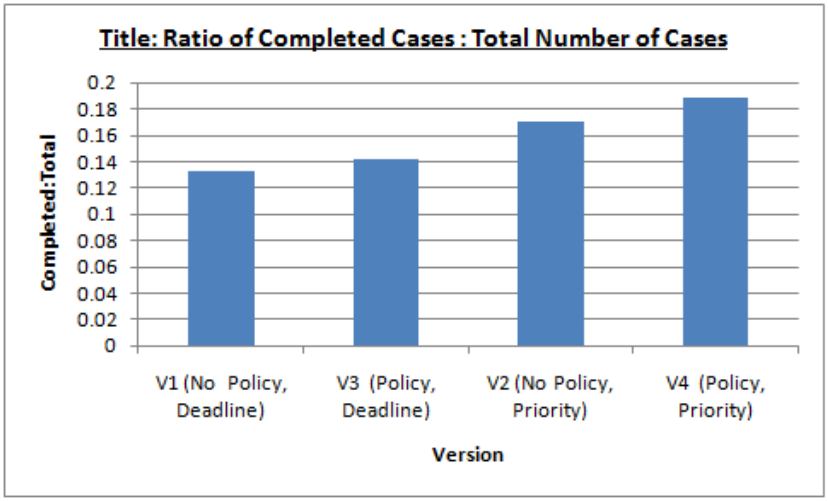


Figure 8: Number of Cases Completed out of the Total Received

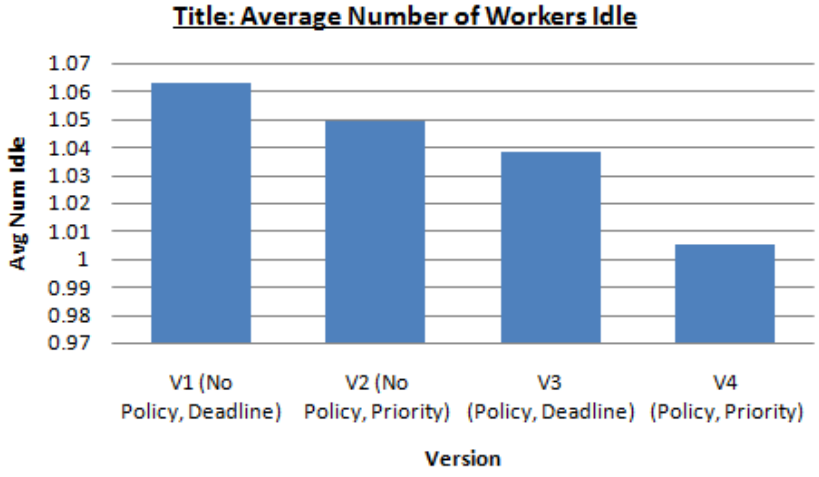


Figure 9: Number of Workers Idle out of 5 in dispensary

7 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 the use of assignment policies and different scheduling protocols (i.e. by priority or deadline).

Our results illustrated that the use of assignment policies and scheduling by priority would improve the business process. These quantitative findings suggest various improvements that can take place in a busy pharmacy, such as the use of priority queues and procedures put in place to enforce that the workers should only take on roles down the role hierarchy if no other capable worker is available.

In the future, we plan to continue to explore the stochastic behaviour of the model by comparing the effectiveness of load balancing and escalation handling . This method can also be further evaluated in terms of scalability will be 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.

[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.

[EPS73] H. Ehrig, M. Pfender, H. J. Schneider. Graph-Grammars: An Algebraic Approach. In *FOCS*. Pp. 167–180. IEEE, 1973.

[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.

[HEGH11] R. Heckel, H. Ehrig, U. Golas, F. Hermann. Parallelism and Concurrency of Stochastic Graph Transformations. *Aussois IFIP WG1.3 meeting*, January 2011.

[HT10] R. Heckel, P. Torrini. Stochastic Modelling and Simulation of Mobile Systems. In Engels et al. (eds.), *Graph Transformations and Model-Driven Engineering*. Lecture Notes in Computer Science 5765, pp. 87–101. Springer Berlin / Heidelberg, 2010.

[HW95] R. Heckel, A. Wagner. Ensuring Consistency of Conditional Graph Grammars - A Constructive Approach. In *Proc. of SEGRAGRA'95 "Graph Rewriting and Computation", Electronic Notes of TCS*. ENTCS 2, pp. 118–126. 1995.

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

[KOD10] A. Karatas, H. Oguztüzün, A. Dogru. Global Constraints on Feature Models. In Cohen (ed.), *Principles and Practice of Constraint Programming CP 2010*. Lecture Notes in Computer Science 6308. Springer Berlin / Heidelberg, 2010.

[ML05] T. von der MaSSen, H. Lichter. Determining the Variation Degree of Feature Models. In Obbink and Pohl (eds.), *Software Product Lines*. Lecture Notes in Computer Science 3714, pp. 82–88. Springer Berlin / Heidelberg, 2005.

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

[(OM09] O. M. G. (OMG). Business Process Modeling Notation (BPMN). Technical report Version 1.2, January 2009.

[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.

[SGB81] D. G. Shimshak, D. Gropp Damico, H. D. Burden. A priority queuing model of a hospital pharmacy unit. *European Journal of Operational Research* 7(4):350–354, 1981.

[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*.