An Enactment-Engine Based on Use-Cases

Avner Ottensooser and Alan Fekete

School of Information Technologies, University of Sydney, Australia
{avner,fekete}@it.usyd.edu.au

Abstract. We show how one can control a workflow enactment engine based on the information which is available in written use cases (as produced by requirements elicitation). We give details of how different aspects of the engine can be configured, including the process definition, workflow participant profiles, user interface, audit data, *etc*. These techniques have been carried out in an industrial setting, with considerable success. Our methods are applicable to engines for business process management, web service orchestration, and traditional workflow.

1 Introduction

The automation of enterprise activity has been a major trend in recent decades, and a key enabler for this trend is the widespread adoption of engines that allow the computational management of processes. These engines were much studied in the 1990s under the term "workflow management", and later a wider horizon has been represented as "business process" execution engines; most recently there has been much attention given to "orchestration for composite web services" which involves the same ideas in the context of business-to-business integration. In this paper we will speak of workflow, but the ideas apply equally in all these settings.

A key feature of any enactment engine is a format for defining the processes that will be executed. Many proprietary languages have been used in commercial products, standards have been proposed, and many more research papers have been written. The most widespread approaches have their roots in a graph or network models, and can be formalised with Petri Nets or similar representations. For example, some vendors of industrial workflow engines, such as IBM and TIBCO, deploy dialects of the UML activity diagram to configure their workflow engines. Other proposals have been based on event-condition-action rules. All these approaches, however much they differ in details, depend on a workflow configuration officer producing a model or definition of each process in a special format, for the purpose of controlling the execution in the enactment engine.

In this paper, we propose a different approach. Rather than asking the workflow configuration officer to model the business processes in a special format, we make use of a well-accepted format for eliciting system requirements: the written use cases [3], which are commonly produced during the requirements gathering stages of projects. We conjecture that the use cases contain the information needed to configure the enactment engine. A great advantage of our approach is the reduction in effort by the workflow configuration officers, who can re-use artifacts business analysts usually generate, instead of undertaking a separate step to analyse and model the business processes. As well, there is good evidence that use cases are an effective means for communicating with domain experts. Use cases seem to scale well, in contrast to say UML Activity Diagrams which become very crowded on realistic domains.

Here is a brief overview of our approach from the point of view of its users; much more detail is given in section 3. With use cases on hand, the workflow administrator creates routing sheets, each describing several action steps which should be performed as a group. When a work item arrives, the first workflow participant to touch the work item begins by cataloguing the work item, and then pilots[1] the work item's flow through the organisation, by linking routing sheets with the work item. Following this, each activity is done by a workflow participant, who continues to perform activities according to the routing sheets, until eventually an activity is found that the participant can't deal with, at which point the work item is passed to another participant. As each activity is executed, the participant acknowledges this to the system. From time to time the workflow system records audit data describing the work item's attributes and progress.

The techniques discussed in this paper have been exploited in commercial practise by the first author at BT Financial Group, an Australian financial services enterprise that is now a subsidiary of Westpac Banking Corporation. BT Financial Group developed a unique Workflow Enactment Engine, whose details were configured as described above. This engine was configured manually from the use cases by business analysts, and the engine ran on top of a conventional workflow engine. One can also imagine similar ideas used in other settings, for example, the configuration might be generated automatically (with suitable natural language processing), or the enactment could be done using a standard relational dbms.

In Section 2, we summarise related work, introduce the terminology, and describe in detail the components of workflow engines. In Section 3, we explain how each configuration component is inferred from the written use cases. Section 3.6 presents some enhancements on the basic idea of use-case-based configuration. Section 4 describes our larger research agenda and its progress to date. In section 5 we conclude with a report on our experience at BT Financial Group, and our reflections on this.

2 Background, Use Cases and Workflow

Here we point to some of the most closely relevant or seminal work. There is of course far more published on both use cases and workflow management than we can cite; more citations can be found in expositions such as [16] and [26]. We then describe the key ideas we build on, as a way of fixing the terminology used in this paper.

[1] We use the word as in a maritime pilot, who helps a ship follow the correct path.

2.1 Previous Related Work

This paper stands on the shoulders of two research communities: Requirements Engineering within Software Engineering research, and Workflow (later called Business Process Management) within Databases and Information Systems. In one of the few researches we found that bridges the two communities, Lee *et al* show that use cases can be transformed into Petri Nets [15].

Use cases were proposed by Jacobson [13] and a diagramming notation for them was included within the UML standard in 1997. The use case technique, arguably one of the best and most widely employed requirement gathering techniques in the industry, is accepted by both IT professionals and business managers [16, Page 297] [4]. Of the 28 dialects of use cases Hurlbut surveyed [12] we adopted the written one described by Cockburn [3]. There are various guidelines for expressing requirements in use cases [5]. Use cases have been found to be effective for generating test suites [7], and for generating security policies [9].

Since the 1990s, the Workflow community researched many different approaches to defining and enacting business processes, and many research prototypes and commercial products have embodied these. The community has an industry body, the Workflow Management Coalition [28], which provided a reference model [11] and terminology [29] which we adopted. There is also a rich pattern library [30, 21] which we use elsewhere to evaluate our method [19].

By far the most popular category of process definition languages uses a visual presentation based on a graph, which connects activities in the order they must be carried out, with connectors representing decision branches, parallel forks, merges *etc.* For example, the UML activity diagram has become widespread for informal modelling, and it is also accepted as an input notation in some engines [22]. The underlying theory for all these graph-based approaches can be expressed in terms of Petri Nets, and some proposals have even adopted variants of the Petri Net directly as a notation. Van der Aalst summarised and evaluated the research [24, 25]. Particular virtues of Petri Nets include their support for automated analysis [1], such as checking for deadlocks [8]. Another class of workflow description languages is based on Event Condition Action rules [6]. A recent example, focusing on service oriented systems, is [17]. Casati *et al* [2] show how to convert graph-based definitions to rules, and provide models for the relational structures we have used to store descriptions within our engine. Non-functional properties such as performance and cost have also been studied [18].

2.2 Use Case

The use case model is an illustrative, incremental requirements elicitation tool that uses *Actors* and *Use Cases*. *Actors* illustrate what exists outside the system and interacts with it. An Actor may be a person in a role (eg Customer), or it may be an external system (eg a credit rating agency) or even a device. *Use Cases* describe what the system is supposed to do. A use-case illustrates what is (or will be) in the system by following a specific, yet complete, flow in the system [13, Section 6.4]. According to Cockburn [3], each use cases describes, in a controlled

```
Use case name: Apply to invest money in a fund

Main success scenario:

1. The mail room scans the application form to the imaging system.   Order = 1
2. The data entry person keys the deposit to the system.             Order = 2 ! Parallel to 2a
3. The system sends transaction confirmation to the investor.        Order = 6
4. The process ends.                                                 Order = 7

Extensions:

2a. The application is for more than AU$1,000,000                             ! AND-Split
    2a1. The Senior Data Entry Person also keys the deposit.         Order = 2 ! Parallel to 2
    2a2. The system reconciles the two data entries.                 Order = 3 ! AND-Join
    2a2. The flow continues at line 3

    2a2a. The reconciliation failed                                            ! OR-Split
          2a2a1. The system sends the two data entries to the         Order = 4
                 senior data entry.
          2a2a2. Senior data entry corrects the data in the system    Order = 5
          2a2a3. The flow continues at line 3

2b. The form arrived unsigned.                                                 ! OR-Split
    2b1. The Data Entry Person calls the Investor, requesting
         a signed form.                                               Order = 2
    2b2. The current process ends (the signed application
         will restart the process).                                   Order = 3
```

Fig. 1. The sample business process used in our paper expressed as a written use case

natural language (English phrases), the interactions between users who wish to achieve a goal and a system, and thus it functions as a contract for the behaviour of the system in meeting the goal. A written use case mentions: (1) Actors such as humans or systems who interact with the system. (2) What must be true in the system for the interactions to be feasible. (3) A main success scenario (happy day scenario) and alternative scenarios, that indicate how the interaction with the system occurs when every thing goes well. And (4) extensions which indicate an abnormal, incorrect, or otherwise unusual situation. *E.g.*, in (Figure 1), the success scenario is steps 1 to 4, while 2b is an extension which is applied when an investor forgot to sign a form.

When we wish to change the system's behaviour, we remodel the appropriate *Actor* and *Use Case*. With the changed *Use Case Model* in hand, we change other models (object, analysis, design , implementation and testing). When we design systems this way the system model will be *Use Case Driven* [13, Section 6.4.2]. In this paper we show how one can configure an enactment engine directly from the written use-case. The following use case is the example we use throughout this paper, it follows the written use-case as described in [3] except that we add an extra attribute "Order" to each step. The order is closely related to the step number, but it allows for indication of when steps can be done independently.

2.3 Workflow

Workflow is the automation of a business process, in whole or part, during which information and work lists are passed from one participant to another for action, according to a set of procedural rules [29, Page 8].

In early times, computer applications were designed to fully support several business transaction types. Administrators invoked applications once all required information was at hand, and processed transactions from start to end, each in a single iteration. In the 70s image management emerged, creating queues in front of administrators who pulled work from the queues and processed the work sequentially. Today, the processing of business transactions is spanning multiple systems, by multiple specialised organisational role bearers, as data drips into the organisation(s). This style of business processing is supported by workflow engines. A *Workflow Engine* is a generic software system driven by explicit process design to enact and manage operational business process [21]. A *Workflow Management System* defines, creates and manages the execution of workflow through the use of software, running on one or more workflow engines, which is able to interpret the process definition, interact with workflow participants and, where required, invoke the use of IT tools and applications [29, Page 9].

Here we describe, using figure 2, the data the workflow administrator gathers, when configuring the enactment-engine which we have used in BT Financial Group. In section 3 we describe an algorithm we used to infer this data from written use cases which were produced during requirements elicitation. In section 4 we outline further research, in which we will explore the wider relevance of this approach.

Our data model is shown in figure 2. At its core resides the business transaction routing sheet. Like the routing documentation used in production floors to describe the production processes an order has to pass, the business transaction routing sheet explains the activities that have to be executed as a group to fulfil part of the use case.

A business transaction routing sheet has one or more activities [29, Page 12]. To each activity one organisational role is assigned [29, Page 54], and an attribute named “order”. As will be described later, the “order” attribute is instrumental in handling parallel work.

Each routing sheet has an attribute named “observation” that the workflow engine uses to build a menu from which the pilot links routing sheets with work items. Each activity has an attribute named “instruction” which the workflow engine uses to prompt the workflow participant to execute the activity.

The details of the workflow participants [29, Page 17] are stored in a table, and another table describes the proficiency of workflow participants in various roles. These two tables together are the workflow participant profile, and one may be right to assume that the profile is populated by human resources.

The last data element is created at run time. We named it “run time construct”. At its core resides the work item [29, Page 18]. The work item is logically linked to one or more routing sheets. Physically we link it to all the activities that constitute these routing sheets. At run time, the workflow engine groups one or more activities into a worklist [29, Page 19] and dispatches this to a workflow participant. A “worklist” is the atomic unit of work dispatched to workflow participants (indeed the activities that combine to form a worklist may come from different work items or even different business processes).

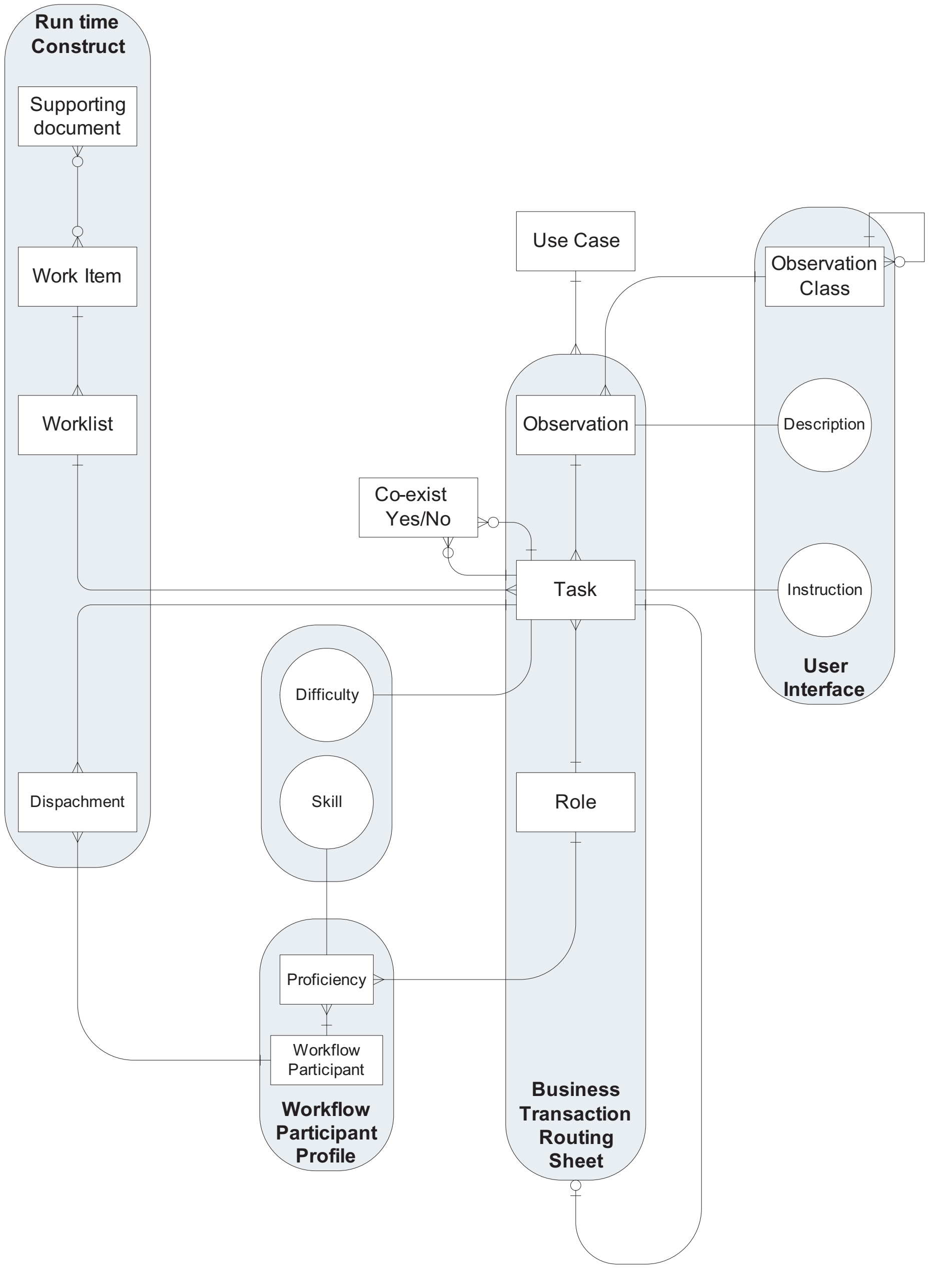

Fig. 2. Use Case Oriented Data Flow Engine — Data Model

A work item is associated with supporting documents. In older times, when imaging systems were promoted as workflow engines, one supporting document was equivalent to one work item; a 1 : 1 ratio. Nowadays, any ratio is acceptable. $0:1$ ratio exists when a work item is not initiated by a document, *e.g.* periodic review. $n:1$ ratio exists when several documents are required for a work item to complete. $1:n$ ratio exists when one document initiates several work items, *e.g.* when employer sends the details of several pension payments in one spreadsheet. We even observed $m:n$. The nature of supporting documents evolved as well, from paper to XML. Here we made an exception to our policy of adhering to the Workflow Management Coalition Terminology by selecting the term "Supporting Document" over the coalition's "Workflow Relevant Data" [29, Page 45].

3 Inferring Workflow Engine's Configuration from Use Cases — An Algorithm

Having introduced the use case terminology and the workflow data model, we now show how workflow administrators in BT Financial Group infer its content from use cases written in Cockburn's notation [3]. In section 4 we conjecture that the algorithm described can be followed in general.

3.1 Order of Processing

The only change we had to make to the use case dialect of Cockburn, as documented [3], is to add the order of processing identifier to action steps. In general, the order of processing identifier is a monotonically increasing integer. However, if the order of some action steps is of no importance, or the action steps follow an AND-Split [29, Page 30], then these action steps share an order processing identifier. *E.g.* in figure 1 above, action steps 2 and 2a1 share '2' as the order of processing. AND-Joins [29, Page 31] are described by an action step whose order of processing identifier is bigger than that of the parallel action steps. *E.g.* in Figure 1 above, action step 2a2 with '4' as order of processing, joins steps 2 and 2a1. If several streams of activity start, then the streams are represented by sub-use cases (Sub Process [29, Page 27]), each represented by a single action step. In our technical report [19] we offer use-case descriptions for all 43 workflow patterns identified by the Workflow Patterns Initiative [30].

In BT Financial Group, the business specific, dispatcher related data elements were: value date, product, client pressure, value, and distribution channel.

3.2 Inferring the Process Definition

A process is described by a set of individual business transactions routing sheets. In our example (Figure 3), four individual business transactions routing sheets are configured. The main success scenario, each alternate flow and each extension are all the observation elements in individual business transactions routing sheet. Each group of action steps that follows them is the reoccurring activity element.

To infer the role, and the activity, from the use-case step, is easy because the use case structure clearly defines who does what. According to Cockburn

```
Business-Transactions Routing Sheet 1

    Observation     = Small Application ! (Or default flow)

       Activity     = Scan the application form
           Role     = mail room
          Order     = 1

       Activity     = Key into the system
           Role     = Data Entry Person
          Order     = 2

       Activity     = Send transaction confirmation
           Role     = The system
          Order     = 6



Business-Transactions Routing Sheet 2

    Observation     = Application Bigger than AU$1,000,000.00

       Activity     = Key into the system
           Role     = Senior Data Entry Person
          Order     = 2

       Activity     = Reconcile the two data entries
           Role     = The system ! Automated process
          Order     = 3 ! If order was 1, the dispatcher would be
                        ! able to dispatch the two Activities in parallel,
                        ! something that may be sensible.



Business-Transactions Routing Sheet 3

    Observation     = Application with a missing signature

       Activity     = Call the investor requesting a signature.
           Role     = Senior Data Entry Person
          Order     = 99 ! any number will do as this is a fatal error.



Business-Transactions Routing Sheet 4

   Observation      = The reconciliation failed

       Activity     = Send the two data entries to the senior data entry.
           Role     = System
          Order     = 4

       Activity     = Correct the data
           Role     = Senior Data Entry Person
          Order     = 5
```

Fig. 3. The routing sheets for our sample business process (Figure 1)

[3, page 90] the action step structure should be absolutely simple, as shown in Figure 4. The role for the activity is the grammatical subject in the step, and the activity is given by the verb and any grammatical direct or indirect objects.

```
Subject... Verb... direct object... prepositional phrase.

The data entry person ... keys ... the deposit ... to the system.
```

Fig. 4. The syntax of an action step

3.3 Inferring the User Interface Specification

In this section we articulate the basic constructs needed to apply the use case notation to the configuration of a workflow engine's run time user interface. At run time, workflow participants request the next work item (GetNext). The dispatcher, a component of workflow engines, matches the workflow participant's roles and skills with the work in the queues and assigns a work item to the workflow participant. The workflow participant then:

Catalogue – When a work item arrives, the first workflow participant to touch the work item catalogues the work item by assigning to the work item attributes such as the business process which the work item must follow, the customer identifier (in BT Financial Group, that is the point where new customers are keyed into the systems), as well as business specific dispatcher related information. To catalogue work items, BT Financial Group uses a key combining the customer's identifier, the business line and the transaction type. Supporting-Documents are catalogued in BT Financial Group using monotonically increasing, non contiguous integers, with a check digit concatenated. An XML document is usually catalogued by a computer programme.

Pilot – As in a maritime pilot, pilots describe the route the work item will pass through the organisation by linking routing sheets to work items using a classified menu of observations (more on the classification structure in section 3.6). At this stage work items can be spawned or merged, and supporting documents, that arrived previously, can be attached to the work item. A pilot can be a computer programme or a human. At every stage in the life cycle of the work item, a workflow participant may further refine the piloting of the work item.

Execute – The pilot may execute the work, or leave it to a specialised workflow participant. At this stage the workflow participant requests the next work item (Get Next) and the workflow engine dispatches a work item, the cataloguing attributes that was previously assigned, the supporting document(s) and, according to the observations the pilot had prescribed, a list of activities the processor is expected to perform. The workflow participant then performs the appropriate activity on the work item (in Figure 1, key it to the mainframe, verify it, or contact the customer).

Acknowledge – Following the execution of each activity the workflow participant flags it as "Done", and other activities as "Diarised" or "Should not be done", until all Activities are completed. Following the acknowledgement, the workflow participant may either request the next work item or terminate the session.

3.4 Inferring the Audit Data Terminology

In BT Financial Group, the Workflow-Engine uses event-driven asynchronous messages to communicate with an external reporting engine by sending instances of the following three message classes:

Work Item Message – Sent when a work item is created, spawned, re-catalogued, terminated, or, merged into anther work item. Records a time stamp and the cataloguing information (described in section 3.3 above). Used to monitor adherence to external Service Level Agreements.

Observation Message – Sent when an Activity is assigned to a work item. Records properties of a work item. Used for quality assurance (*e.g* in Figure 1, how often do investors forget to sign application forms).

Worklist Message – Sent when a Worklist is queued, starts, ends, or diarised. Records who performed the activity and how fast. Used to monitor adherence to internal service level agreements.

3.5 Inferring the Content of the Dispatching Queue

The dispatcher watches two lists: (1) available workflow participants with roles and skills; and (2) piloted activities, with roles, difficulty and other configurable dispatching parameters. These two lists are used to implement dispatching patterns that are only limited by the imagination.

Thus, in this section we have demonstrated how the content of the data model can be inferred from use cases.

3.6 Extensions

This section describes further refinements to Cockburn's use case notation [3], which we have found to be useful.

Skills and Difficulty – To increase the granularity of the dispatching of Activities to Role bearers, difficulty was assigned to activities and Skill was assigned to Role bearers. The dispatcher is configured to assign Worklists to Role bearers who are sufficiently skilled to handle the most difficult Activity in the Worklist.

Activity co-existence – To increase piloting quality (see section 3.3 above), the Workflow configuration officer may articulate whether certain activities may co-exist with other activities.

Observation Menu – To ease the task of locating observations, a category based tree structure was implemented.

3.7 Implementation of the Use Case Model on Off the Shelf Workflow Engine

While the ideas expressed above can be implemented on a relational database, users may find it beneficial to implement the model on top of commercial Workflow Engine, as happened in BT Financial Group where FileNET software was used. In that case it is recommended to connect all workflow participants to each other and to direct the flow using the dispatcher described above. In that case, the system should be configured as having only three queues.

Unpiloted Work – As its name suggests, that queue will hold all work items that are not catalogued. As cataloguing all incoming document may be a priority, this queue may have higher priority than the Work in progress queue.

Work in progress – The queue from which the dispatcher allocates work items to workflow participants.

Completed work – The storage of work that ended.

4 Further Research

We presented an approach which we have applied in one, albeit complex, environment within BT Financial Group. At this point we offer two conjectures.

Conjecture 1. *Use cases written in the Cockburn format [3], with order assigned to action steps, provide sufficient information to describe the workflow in any reasonable system.*

Conjecture 2. *A workflow which is given as a written use-case avoids many of the errors that can arise with general graph-based notations: it has an end, has only reachable nodes, and has no dead ends.*

We plan to explore the validity of the conjectures through several research activities. For the first conjecture, we will start by showing that written use-cases can deal with many of the workflow situations already known. In particular, we have followed the methodology deployed by Russell *et al* for evaluating the richness of UML2.0 activity diagrams [22]; we have shown how to give written use-case descriptions for the standard workflow patterns. Our analysis is in [19]. The second conjecture will be explored by following the research of Lee *et al* [15], and looking at the properties of Petri Net equivalents to use case descriptions.

5 Experience, Evaluation, Discussion and Conclusion

5.1 Experience Gained at BT Financial Group

Starting in 2002, a group at BT Financial Services, managed by the first author, operated a Workflow Management System, supported by a specially-written enactment engine. This system was based on the principles presented in this paper. In BT Financial Group we found that:

– In April 2007, 368 Business-Process were controlled by the system. Each business process had on average 26 possible activities. At that point in time, on a typical day, about 600 administrators were logged in. On average day in April 2007, approximately 10,000 business process instances which were supported by 12,000 images were executed. The number of audit rows generated daily was about 300,000. The administrators were located in three Australian states and in India.

- Use cases became the primary tool for the workflow configuration officers. These officers' productivity was so high that in 2006 Westpac Life, a sister company which is in the life insurance business, migrated its entire processes into BT Financial Group's Workflow-Engine within five weeks.
- Analysis of the audits data collected was instrumental for the identification, quantitative justification, and subsequent quantitative evaluation, of Six-Sigma process improvements programmes.
- Line managers, with general accounting skills, feel comfortable to add, maintain, or remove activities.
- In BT Financial Group, worklists are created whenever a processor requests the next worklist. The approach where administrators requests a work item (get next) and the dispatcher assigns them the most appropriate one, rather then letting administrators "Cherry Pick" work items, increases the management control.
- When BT Financial Group placed skilled personnel as pilots, quality was built from the beginning at the price of overloading experts with mundane activities. When BT Financial Group placed unskilled processors at the beginning, work often arrived to the skilled personnel none the less, but for the wrong reason — repair.
- Some business areas encouraged pilots to pilot and perform the prescribed activity in a single session. Other business areas discouraged this.
- Some business areas tried to complete the piloting early in the morning and process in the rest of the day. Other business lines piloted and processed throughout the day.
- BT Financial group experimented with the following dispatcher patterns:
  - FIFO
  - The hardest job one can do in descending age order.
  - The oldest un–piloted work, then the oldest and the hardest work item a Workflow Participant may perform, from the oldest day.
  - Business related consideration such as priority for redemptions over deposits and of cash transactions over manged fund transactions.
- The large majority of business processes did not have any scope for within-instance parallelism.
- Unfortunately we found that the routing sheet observation attribute was too often identical to the instruction attribute of the activity.

5.2 Limitations

We identified several issues where the use case notation is less satisfactory than other workflow description notations. We discuss these in turn.

Use cases can be *ambiguous* as they use natural language. This seems to be an evident trade-off against the ease of communication with business experts. To overcome the natural language ambiguities, Cox *et al.* [4, 5] show how to improve the quality of use cases with checklists driven reviews. As Törner *et al* suggests [23], it is possible to increase the Correctness, Consistency, Completeness, Readability and level of detail as well as to avoid ambiguity.

There is little direct support for *analysing flaws* in use case descriptions, compared to Petri Net methods which allow automated detection of deadlocks possibility, unreachable nodes and uncontrolled process termination. However, as Lee *et al* [15] have shown, use cases can be considered as a set of interacting and concurrently executing threads, and that use cases can be transformed into a Constraint-Based Modular Petri Nets based formalism [25]. Once converted into Petri-Nets we can (1) identify inconsistent dependency specifications among Activities; (2) test for workflow safety, *i.e.* test whether the Workflow terminates in an acceptable state; (3) for a given starting time, test whether it is feasible to execute a Workflow with the specified temporal constraints [1] ; (4) test the Workflow for possibility of deadlocks [8].

Use case descriptions generally lead to sequential execution, or at best *low levels of parallelism* within each business process instance.

While the use case notation had these limitations, the overall impact on the company was very positive. In the next subsection we reflect on the ways our approach was beneficial.

5.3 Value Proposition

The value proposition of our approach to Workflow-Engine configuration is that it:

– Reduces the amount of effort required to configure workflow engines, by reusing the organisation's investment in use cases. As use cases are ubiquitous in today's business analysis arena, one would expect that the workflow configuration officers would have use cases available before the Workflow configuration commences.
– Audit data and the user interface are maintained as part and parcel of the process definition reducing development effort.
– Allows the two flexibilities that Heinl *et al* [10] required from a Workflow-Engine, namely:

  Flexibility by Selection – the processor has the freedom to choose between different execution paths if necessary.

  Flexibility by Adaptation – it is possible to change the Workflow definition at run-time by adding, removing or altering Business-Transaction Routing Sheets.

– Written use cases provide descriptions which can be understood by various stake holders in a straightforward manner. Cox *et al* suggest that end-users

do understand well written use cases [4]. We have not found corresponding research that suggests that end-users can understand and review Workflow annotated by Petri-Nets, and our experience, together with the discussion in [22], suggests that complex UML2.0 Activity Diagrams are beyond end-users' reach when modelling resource-related or organisational aspects of business process. This approach to workflow has allowed workflow activities to be performed by a wider range of employees. In particular, it is not necessary for participants to understand workflow notations like graphs or activity diagrams; the written use-case can be followed by all employees.
– Our approach enables pilots who are unfamiliar with the underlying routing to make complex routing decisions by concentrating on observations rather then activities.

References

[1] Adam, N.R., Atluri, V., Huang, W.: Modeling and Analysis of Workflows Using Petri Nets. Journal of Intelligent Information Systems 10, 131–158 (1998)

[2] Casati, F., Ceri, S., Pernici, B., Pozzi, G.: Deriving Active Rules for Workflow Enactment. In: Thoma, H., Wagner, R.R. (eds.) DEXA 1996. LNCS, vol. 1134, pp. 94–115. Springer, Heidelberg (1996)

[3] Cockburn, A.: Writing effective use cases. Addison-Wesley, London (1999)

[4] Cox, K., Aurum, A., Jeffery, R.: An Experiment in Inspecting the Quality of use case Descriptions. Journal of Research and Practice in Information Technology 36(4), 211–229 (2004)

[5] Cox, K., Phalp, K., Shepperd, M.: Comparing use case Writing Guidelines. In: Proc. Workshop on Requirements Engineering: Foundation of Software Quality (REFSQ'01), pp. 101–112 (2001)

[6] Dayal, U., Hsu, M., Ladin, R.: Organizing Long- Running Activities with Triggers and Transactions. In: Proc ACM International Conference on Management of Data (SIGMOD'90), pp. 204–214 (1990)

[7] de Figueiredo, A.L.L., Andrade, W.L., Machado, P.D.L.: Generating Interaction Test Cases for Mobile Phone Systems from use case Specifications. ACM SIGSOFT Software Engineering Notes 31(6), 1 (November 2006)

[8] Ezpeleta, J., Colom, J.M., Martinez, J.: A Petri net based deadlock prevention policy for flexible manufacturing systems. IEEE Trans on Robotics and Automation 11(2), 173–184 (1995)

[9] Fernandez, E.B., Hawkins, J.C.: Determining role rights from use cases. In: Proc ACM Workshop on Role-Based Access Control (RBAC '97), pp. 121–125 (1997), DOI= `http://doi.acm.org/10.1145/266741.266767`

[10] Heinl, P., Horn, S., Jablonski, S., Neeb, J., Stein, K., Teschke, M.: A comprehensive approach to flexibility in Workflow management systems. In: Proceedings of the international Joint Conference on Work Activities Coordination and Collaboration (WACC'99) pp. 79–88. (1999), DOI= `http://doi.acm.org/10.1145/295665.295675`

[11] Hollingsworth, D.: The Workflow Reference Model. Document Number TC00–1003 of the Workflow Management Coalition, Document Status - Issue 1.1 (Janu-vary 19, 1995)

[12] Hurlbut, R.: A Survey of Approaches for Describing and Formalizing use cases. Technical Report 97– 03, Department of Computer Science, Illinois Institute of Technology, USA. (1997), Found at
http://www.iit.edu/~rhurlbut/xpt-tr-97-03.html

[13] Jacobson, I.: Object-Oriented Software Engineering. Addison-Wesley, London (1992)

[14] Kim, J., Spraragen, M., Gil, Y.: An intelligent assistant for interactive Workflow composition. In: Proc International Conference on Intelligent User interfaces (IUI'04), pp. 125–131 (2004), DOI=
http://doi.acm.org/10.1145/964442.964466

[15] Lee, W.J., Cha, S.D., Kwon, Y.R.: Integration and analysis of use cases using modular Petri nets in requirements Engineering. IEEE Transactions on Software Engineering 24(12), 1115–1130 (1998)

[16] Leffingwell, D., Widring, D.: Managing Software Requirements: A use case Approach, 2nd edn. Addison-Wesley, London (2003)

[17] Nepal, S., Fekete, A., Greenfield, P., Jang, J., Kuo, D., Shi, T.A: Service-oriented Workflow Language for Robust Interacting Applications. In: Proc International Conference on Cooperative Information Systems (CoopIS'05), pp. 40–58 (2005)

[18] O'Sullivan, J., Edmond, D., ter Hofstede, A.: What's in a Service? Distrib. Parallel Databases 12, 117–133 (2002)

[19] Ottensooser, A., Fekete, A.: Workflow Patterns Represented in Use-Cases, Technical Report Number 611, School of Information Technologies, University of Sydney

[20] Russell, N., ter Hofstede, A.H., Edmond, D., van der Aalst, W.M.P.: Workflow Resource Patterns. BETA Working Paper Series, WP 127, Eindhoven University of Technology, Eindhoven (2004), http://fp.tm.tue.nl/beta/

[21] Russell, N., van der Aalst, W.M.P., ter Hofstede, A.H., Edmond, D.: Workflow Resource Patterns: Identification, Representation and Tool Support. In: Pastor, Ó., Falcão e Cunha, J. (eds.) CAiSE 2005. LNCS, vol. 3520, pp. 216–232. Springer, Heidelberg (2005)

[22] Russell, N., van der Aalst, W.M.P., ter Hofstede, A.H., Wohed, P.: On the suitability of UML 2.0 activity diagrams for business process modelling. In: Gruska, J. (ed.) Mathematical Foundations of Computer Science 1977. LNCS, vol. 53, pp. 95–104. Springer, Heidelberg (1977)

[23] Törner, F., Ivarsson, M., Pettersson, F., Öhman, P.: Defects in automotive use cases. In: Proc ACM/IEEE international Symposium on Empirical Software Engineering (ISESE'06), pp. 115–123 (2006), DOI=
http://doi.acm.org/10.1145/1159733.1159753

[24] van der Aalst, W.M.P.: Three Good Reasons for Using a Petri-net-based Workflow Management System. In: Proc International Working Conference on Information and Process Integration in Enterprises (IPIC'96), pp. 179–201 (1996)

[25] van der Aalst, W.M.P.: The Application of Petri Nets to Workflow Management. Journal of Circuits, Systems, and Computers 8(1), 21–66 (1998)

[26] van der Aalst, W.M.P., Van Hee, K.: Workflow Management: Models, Methods and Systems. MIT Press, Cambridge (2002)

[27] van der Aalst, W.M., Barros, A.P., ter Hofstede, A.H., Kiepuszewski, B.: Advanced Workflow Patterns. In: Scheuermann, P., Etzion, O. (eds.) CoopIS 2000. LNCS, vol. 1901, pp. 18–29. Springer, Heidelberg (2000)

[28] The Workflow Management Coalition can be found, at `http://www.wfmc.org/`
[29] Workflow Management Coalition. Terminology and glossary. Technical Report WFMC-TC-1011, Workflow Management Coalition (February 1999)
[30] The Workflow Patterns initiative is a joint effort of Eindhoven University of Technology (led by Professor Wil van der Aalst) and Queensland University of Technology (led by Associate Professor Arthur ter Hofstede). The publications of the Workflow Patterns Initiative can be found at
`http://www.Workflowpatterns.com/`