Sydney University
Faculty of Engineering and Information Technologies

Doctor of Philosophy Dissertation

Dynamic Workflow-Engine

by

Avner B. OTTENSOOSER

Supervisor: Professor Alan Fekete

Sydney, December 2011

To Tal and Yochi

Contents

Abstract

We present and assess the novel thesis that a language commonly accepted for requirement elicitation is worth using for configuration of business process automation systems. We suggest that Cockburn's well accepted requirements elicitation language — the written use case language, with a few extensions, ought to be used as a workflow modelling language.

We evaluate our thesis by studying in detail an industrial implementation of a workflow engine whose workflow modelling language is our extended written use case language; by surveying the variety of business processes that can be expressed by our extended written use case language; and by empirically assessing the readability of our extended written use case language.

Our contribution is sixfold: (*i*) an architecture with which a workflow engine whose workflow modelling language is an extended written use case language can be built, configured, used and monitored; (*ii*) a detailed study of an industrial implementation of use case oriented workflow engine; (*iii*) assessment of the expressive power of the extended written use case language which is based on a known pattern catalogue; (*iv*) another assessments of the expressive power of the extended written use case language which is based on an equivalence to a formal model that is known to be expressive; (*v*) an empirical evaluation in industrial context of the readability of our extended written use case language in comparison to the readability of the incumbent graphical languages; and (*vi*) reflections upon the state of the art, methodologies, our results, and opportunities for further research.

Our conclusions are that a workflow engine whose workflow modelling language is an extended written use case language can be built, configured, used and monitored; that in an environment that calls upon an extended written use case language as a workflow modelling language, the transition between the modelling and verification state, enactment state, and

monitoring state is dynamic; that a use case oriented workflow engine was implemented in industrial settings and that the approach was well accepted by management, workflow configuration officers and workflow participants alike; that the extended written use case language is quite expressive, as much as the incumbent graphical languages; and that in industrial context an extended written use case language is an efficient communication device amongst stakeholders.

Statement of Originality

In this dissertation I give the first comprehensive report of a research programme I have conducted under the supervision of Professor Fekete. To the best of my knowledge this dissertation contains no material written by another person, except where duly acknowledged. Contributions made to my research programme, and assistance I received in the editing of this dissertation are acknowledged as well. Parts of Chapter 3 – Architecture and Chapter 4 – Case Study were previously published as:

> A. Ottensooser and A. Fekete. An enactment-engine based on usecases. In G. Alonso, P. Dadam, and M. Rosemann, editors, *Proc. BPM'07: Business Process Management (LNCS 4714)*, pages 230–245, Heidelberg, 2007. Springer-Verlag [86].

Parts of Chapter 6 – Readability are currently in press as:

> Ottensooser, A., et al., Making sense of business process descriptions: An experimental comparison of graphical and textual notations. J. Syst. Software (2011), doi:10.1016/j.jss.2011.09.023 [89]

The appendix Participant's workbook and the statistical work at Chapter 6 – Readability were previously published as part of:

> A. Ottensooser and A. Fekete. Comparing readability of graphical and sentential process design notations – data analysis report. Technical report, School of Information Technologies, University of Sydney, Australia, 2010. TR-658, ISBN 978-1-74210-198-9 [88].

The appendix Pattern-Based Analysis of the Control-Flow Perspective of written-use-cases was previously published as:

> A. Ottensooser and A. Fekete. Workflow patterns represented in use-cases. Technical report, School of Information Technologies, University of Sydney, Australia, 2008. TR-611, ISBN 978-1-74210-021-0 [87] .

I present the state of the art — the work of other people, in Chapter 2 – Context. After describing the state of the art I synthesise the state of the art in section 2.6 – Reflections. The rest of this dissertation is my original contribution .

In chapter 4 – Case Study, I describe a workflow engine construction project that BT Financial Group undertook. My role in that project was that of a project director with responsibility for the overall architecture, development, change management, deployment of the workflow engine software and its configuration, the project budget and schedule. Naturally, I harnessed the effort of a development team, a project manager, and change agents drawn from the end user community.

The project was run on strict commercial terms and had no place in it for pure science. Indeed chapter 4 is a study of that single case — a case study by other name. Now, years later, I call upon this case and study it as evidence that a use case oriented workflow engine exists and that it is well accepted by management, workflow configuration officers and workflow participants alike. These conclusions by those in the organisation are not biased since I am no longer involved in the project.

The writing style of this dissertation calls for the use of the subjective pronoun “I” in the reflective sections and the subjective pronoun “we” in all other. The use of the subjective pronoun “we” does not imply that I did not do the work described.

The translations of quotations from the Bible and the Talmud are original.

Acknowledgements

> The wiser Kohellet was, he still imparted knowledge to the people; pondered and researched; and set in order many proverbs.
>
> Ecclesiastes 12:9

I WISH TO THANK my supervisor Professor Alan Fekete for years of constant challenge, support and sensitively administered criticism. In particular I wish to thank Professor Fekete for help in drafting the road map of this research programme and steering my progress along the way. I also thank Alan for shielding me from the mundane chores of university life — the annual progress reports and the ethical approval for our experiment. While Professor Fekete is a mathematician by training, and an original computer scientist, he is above all a teacher, and his calm manner, encyclopedic knowledge, passion for computer science, and to the human interaction within the various computer science communities, attention to the completeness of a line of argument, and insistence on concrete examples will stay with me for many years to come.

I would like to acknowledge the help of the three anonymous reviewers of this dissertation for their contribution to the accuracy, breadth and format of this dissertation.

Special thanks to my beloved wife Dalia Chaia Ottensooser who tolerated the years of effort, energy and attention I have devoted to the research programme; years of effort, energy and attention that should have been deployed to causes much more worthwhile.

During my research programme I had the honour of talking with Professor van der Aalst and the joy of collaborating with Professor Reijers,

Professor Mendling, Dr. Menictas, Professor Judy Kay and Dr. Cockburn on the research presented in Chapter 6 — Readability. I also had the pride and joy of employing my daughter Tal Ottensooser as a research assistant, posting advertisements on notice boards and administrating questioners.

For 12 years I had the honour to observe, and sometimes contribute, to the evolution of the business process management practice at BT. First amongst the numerous colleagues I have to thank is the inspirational Mr. Keith Hallet. Countless planning sessions, in various corners of the globe, long into the night with Keith and his inseparable companion Dr. Walker, J. were the inspiration for the operational concepts of dynamic workflow. I also have to thank Mrs. Sandra Hill for sponsoring the development of a workflow engine in BT, as well as Mr. Tony Forward who authorised the use of experience gained at BT as the basis for this research programme. Special thanks to Mr. Dale Swift who managed the team that developed BTeP, the first use case oriented workflow engine, and reviewed this dissertation.

During my research years I was assisted by Ms. Jenni Michelson and my son Yochi Ottensooser in all English related issues. Both Jenni and Yochi spent countless hours editing my papers and this dissertation, often asking very hard questions, and pointing at important reading material.

During my research programme I was assisted by various scientific tools such as Google Scholar — the portal to the online databases, statistical analysis with the R platform [93]; the LaTeX platform; the BibTeX citation manager; the Memoir class; the Glossaries terms management Package; LEd, a LaTeX editor; YASPER, a Petri-Net simulator [109]; and System Architect, a CASE tool.

Of the material I read during my research programme I was especially inspired by the monograms: *Obedience to Authority: An Experimental View* by Milgram [73]; *Human Computer Interaction* by Dix *et al* [34]; *Writing Effective Use Case* by Cockburn [24] and *Patterns for Effective Use Cases* by Cockburn *et al* [13]; *Object-oriented software engineering* by Jacobson [57] and *Experimentation in software engineering: an introduction* by Wohlin *et al* [116].

While writing this dissertation I always referred to the online edition of the Oxford English Dictionary [25] as a first port of call for definitions, in particular: "Gantt chart", "work", "work flow", "specification", "to gild the lily", "instruct', "observe", "pilot", "induction", "deduction", "placebo' as well as for the lovable English term "Schadenfreude" and the Australian term "larrikin".

Glossary

One of the challenges in a multi disciplinary work, such as this dissertation that draws upon work of the Business Process management, Requirement Engineering, HCI community's work and Linguistics communities, is the bridging of the terminologies of diverse communities. This dissertation is founded upon 78 terms. While I use most of the terms in the definition of other terms, I avoid circular definitions. However, due to editorial considerations, I present in chapter 1 some yet to be defined terms. To help the reader gain quick access to these definitions I use the hyper–linking functionality provided by the GLS LaTex package. The reader who is after an orderly introduction of terms, starting from basic principles sourced from the Oxford English Dictionary, via terms sourced from the professional literature to original terms such as use case oriented workflow engine; should read chapter two first, then chapters three to six and only then chapters one and seven. To help the reader navigate through the terminology maze, I placed an extended glossary section right at the beginning of the dissertation. The glossary defines each term, points to every where page the term is used, and to the terms that rely upon that term. Due to the large numbers of terms introduced in this dissertation, I made the editorial decision not to use acronyms. The style of writing adopted in this dissertation called for a single bibliography entry to the OED. As I had to state the date I accessed each definition online, I attached the date to the quote, otherwise I would have had to place the dictionary 18 times in the bibliography.

absolute readability ($QSet_n$) , n. The measured knowledge of a participant after reading an artifact. 139, 140, 145

action step , n. A labelled written use case line which has a restricted grammatical structure [24, Page 90]. 21, 35, 36, 40–43, 56, 62–64,

should not be done , method. A public method used by the workflow participant to instruct the workflow engine to cancel an activity already prescribed to a work item. 77, 85, 86, 89, 90, *see also* done, workflow engine, workflow, workflow participant, activity, instruct & prescribe

sound workflow net , n. Workflow net $PN = (P, T, F)$ is sound *iff* (i) for every state M reachable from state i, there exists a firing sequence leading from state M to state o $(\forall_M(i \xrightarrow{*} M) \Rightarrow (M \xrightarrow{*} 0))$, and (ii) state o is the only state reachable from state i with at least one token in place o $(\forall_M(i \xrightarrow{*} M \wedge M \geq o) \Rightarrow (M = o))$, and (iii) there are no dead transitions in (PN, i) $(\forall_{t \in T} \exists_{M,M'} i \xrightarrow{*} M \xrightarrow{t} M')$ [3]. 19, 25–27, 57, 116, 120, 121, 124–126, 129, 131, 132, 162, *see also* workflow & workflow net

specification , n. Specific, explicit, or detailed mention, enumeration, or statement of something [25]. x, 41

stakeholder , n. Someone or something that has a vested interest in the behaviour of the use case [24, page 53]. 2, 5, 32, 39, 40, 59, 60, 63, 87, 90, 111, 148, *see also* use case

structurally bounded , a. A Petri net is structurally bounded if the net is bounded for any initial state. 26, *see also* Petri net & bounded

supporting document , n. External artefact that corroborates a business transaction. 70, 74, 85, 94–97, 99, 100, 107

Theseus tree , n. A tree whose root is the start state of a graph S, the nodes are the set of reachable states from S, and the children of a node N are states that can follow N in one step and have not been included in the tree at the same or higher level. 124, 125, 127, 128, *see also* reachability graph of a net

to gild the lily , v. To embellish excessively, to add ornament where none is needed [25], such as adding this definition. x, 67

usability , n. The extent to which product can be used by specified users to achieve a specified goal with effectiveness, efficiency and satisfaction in a specific context of use [58]. 21, 44, 46, 64, 67, 90, 132

work flow , n. in an office or industrial organization *[sic]*, the sequence of processes through which a piece of work passes from initiation to completion [25, accessed 19-March-2011]. x, 6, 29, *see also* work

work item , n. The representation of the work to be processed by a workflow participant in the context of an activity within a process instance [117, Page 19]. 5, 7, 8, 19, 30–32, 62, 63, 73, 74, 77, 83, 85, 86, 88, 89, 95, 97–107, 109, 112, 131, *see also* workflow, workflow participant & activity

workflow , n. 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 [117, Page 8]. 2, 5–7, 9, 10, 15, 18, 21, 25, 27, 29, 30, 87, 88, 97, 98, 105, 111, 113, 115, 117, 120, 121, 125–127, 129, 131, 132, 134, 137–139, 143, 144, 160–163, *see also* work flow & work

workflow configuration officer n. A person who is responsible to feed workflow models into workflow engines. vi, viii, 2, 7–9, 33, 34, 62, 64, 66, 67, 72–74, 80, 81, 87–89, 108, 110, 111, 114, 134–136, 159–161, *see also* workflow engine & workflow

workflow engine , n. A generic software system driven by explicit process design to enact and manage operational business process [95]. v, viii, 6–9, 29–34, 56, 59, 60, 62–64, 66, 70, 72, 74, 77, 83, 85, 86, 88–90, 93, 97, 98, 100, 104, 106, 108–111, 113, 114, 133, 160, 161, *see also* workflow

workflow modelling language , n. A format for defining a process enacted by a workflow engine. v, 2, 6, 7, 9, 11–13, 29, 30, 34, 43, 56, 57, 59, 64, 71, 77, 88–90, 105, 116, 133, 134, 160–164, *see also* workflow engine & workflow

workflow net , n. A Petri net that has one input place i and one output place o, and where for each transition t there is a path from i to o via t. Formally a Petri net $PN = P(P, T, E)$ is a workflow net *iff* (i) there is one source place $i \in P$ such that $\bullet i = \phi$, there is one sink place $o \in P$ such that $\bullet o = \phi$, and (ii) every node $x \in P \cup T$ is on the path from i to o [3]. 19, 25, 26, 117, 120, 124, *see also* Petri net & workflow

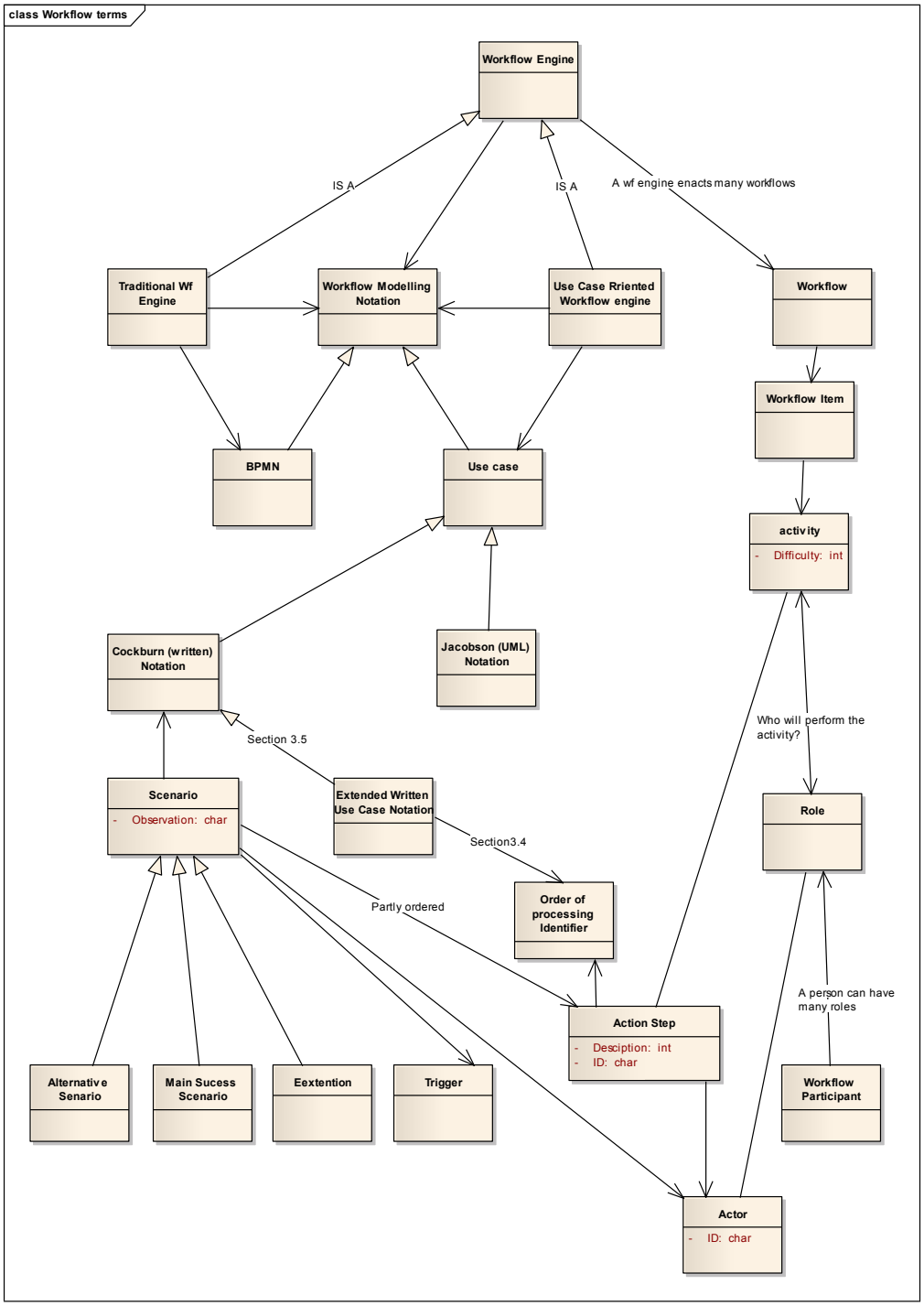


Figure 1 – A reference model of the terminology introduced in the glossary

Chapter 1

Introduction

> There are four languages worth using, Greek for song, Latin for war, Syriac for lamentation and Hebrew for speech.
>
> Rabbi Jonathan from Bait Govrin [33, 43]

Formally articulated, optimised and followed business processes are one of the main assets of contemporary organisations, parallel in importance to franchises, brands, patents, capital and human resources. Some organisations, challenged to continuously improve their business processes, call upon systems that automate the management of the multitude of business processes they perform day in and day out. These systems, in turn, call for dedicated modelling languages for the configuration of business processes. These languages are the subject of this dissertation.

In this dissertation we present and assess a novel approach for the configuration of business process automation systems. Our thesis is that a language commonly accepted for requirement elicitation is worth using for configuration of business process automation systems.

Our research builds upon ideas drawn from the work of several communities. Using ideas sourced from the requirement engineering community, we describe the language we propose for the configuration of business process automation systems. Using ideas sourced from the software engineering community we articulate an abstract architecture of an implementation that automates workflows described in this way and we describe the pro-

cess management practises in an organisation that applied our approach. Using ideas sourced from the business process management community we explore the gamut of business process patterns to which our approach may be applied. Lastly, using ideas drawn from the human computer interaction literature we measure the extent to which a wide cohort of stakeholders can understand our language in comparison with the incumbent language.

In this introductionary chapter, we contrast two approaches to the management of business processes within the organisation — project management and workflow management. We describe engines used for the management of projects and engines used for the management of workflow, and in particular we describe the engines' respective modelling language. Thus we put into context the subject of this dissertation — a workflow modelling language. We then suggest to reuse the written use cases, artifacts that business analysts routinely produce for requirement elicitation, for the purpose of workflow definition.

By reusing the written use cases, rather than translating them into a dedicated workflow modelling language, organisations may improve communication between different stakeholders, including workflow participants, business analysts, modellers, workflow configuration officers and process improvement experts. Reuse may also enable organisations to remove risky translation activities from workflow configuration projects, decrease time to market and reduce costs.

We conclude this chapter by outlining the structure of the research programme on which we report in this dissertation. In that research programme we have undertaken to see how well a use case language could be used as a workflow modelling language.

1.1 The Project Management Approach

Contemporary organisations often apply project management methodologies to projects — the once off medium to large scale activities that have clear objectives, defined start and defined end. To describe projects formally some project managers use Gantt charts. The Gantt chart project modelling language was first published by Henry Gantt in 1903 while working on shop floor optimisation with Frederick Winslow Taylor [114].

> **Definition §1.1**
> Gantt chart, n. The name of Henry Laurence Gantt (1861-1919), American management consultant, used attrib. to designate a

> chart in which a series of horizontal *[sic]* lines shows the amount of work done or production completed in certain periods of time in relation to the amount planned for those periods [25, accessed October 2010].

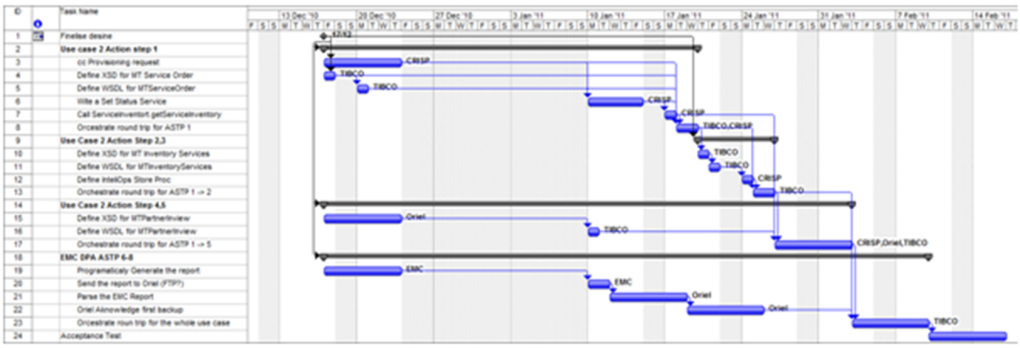


Figure 1.1 – **Gantt chart** — In this example we see a simple project plan that includes 24 activities and four participants. As the figure was drawn from a real life commercial project, fonts were deliberately scaled down to an unreadable size.

With a Gantt chart such as in Figure 1.1, project managers list project activities on the Y axis and time line on the X axis. Horizontal bars represent activities. The horizontal span of an activity represents its time, and arcs that connect these horizontal bars represent dependencies between activities [59, Page 557].

Some project management engines ask project managers to key various properties of project participants such as percent availability or hourly rate, and properties of activities such as the project participants who will perform the activities and their duration. By analysing Gantt charts, some project management engines track circular referencing — the dependency of an activity upon activities that are dependent on it, track resources that are over utilised, and identify the project's critical path — the sequence of activities that any delay in their start or their end will postpone the completion of the entire project. Some project management engines can identify each activity's earliest possible start and latest optimal end, as well as the load imposed on each of the projects' participants, and calculate costs associated with the project [10].

The project management approach scales up well. When an organisation commissions a very large project, a project director may split a handful

of activities within a master plan into independent, manageable, and somehow coordinated activities. The project director may then commission other project managers to describe each of these activities in independent Gantt charts and so on, until the Gantt charts are sufficiently detailed. Examples of very large projects that were managed this way are the invasion of Normandy, the Manhattan project [78, Pages 11-17] and the Apollo program [78, Pages 49-59].

As the Gantt chart language has the "AND split" and the "AND join" building blocks, Gantt charts can record dependencies between activities. There are, however, some limitations to the expressive power of the Gantt chart language. Gantt charts can record only one scenario, most usually the optimistic one, we call it the main success scenario. A single Gantt chart cannot record a scenario and alternative scenarios as the Gantt chart language lacks "OR split" and "OR join" building blocks. Gantt charts, thus, cannot express flow controls such as "*if... then... else...*", "*pick one of several*", or "*repeat... until*". Rather, project managers cater for risk — the unpredictability inherent to many activities, by adding a safety margins to the duration of activities and by recording all imaginable deviation from the main success scenario on a secondary, manually connected aftefact — the risk register. This is true for both PRINCE2 and PMBOK.

> **Definition §1.2**
> risk register, n. A project management artefact used to record possible deviation from the project plan, combined with the identification of procedures to avoid or minimise the impact of such deviation [85, Section 8.3.4].

Project managers record in the risk registers as many deviations from the main success scenario as imaginable and their estimated probability. For each possible deviations from the main success scenario, project managers use risk registers as means of recording countermeasures that are designed to prevent the deviation, activities that will take place should the deviations from the main success scenario occur, and status of the risks [85, page 356].

1.2 The Workflow Management Approach

To use an analogy, a project manager would treat an activity as an elegant Bond street tailor would treat a three piece £5,000.00 suit. The provisioning process in Bond street usually involves four meetings with the client, first

for initial measurement and garment choice, second measurement of the unfinished suit, optional third that may yield some minor alternation, and fourth for delivery.

When the customer, though, is after 15,000 different T shirts, at $0.50 a piece, Bond street's four appointment process does not scale down well. At $0.50 a piece, the Bond street process cannot be economically applied to each and every T shirt. Similarly the project management approach is not suited to manage frequently invoked business process.

The project management does not economically scale down to handle the multitude of repetitive work items that the organisation processes day in and day out. These repetitive work items can be expensive and complex, and would have required dedicated management structure, had they not been so common and so vastly numerous.

To further contrast the project management approach and the workflow management approach, please look at the different order of magnitudes in the following examples: BT Financial Group, the financial services arm of Westpac Banking Corporation, started on an average day in April 2006, approximately 10,000 work items (or business process instances). Example of these business processes are "managed funds — initial deposit", "retirement product — additional application" "switch", "transfer", "cash redemption" or "personal detail amendment". In comparison, in April 2006 BT Financial Group started less than 20 projects — a six orders of magnitude difference.

Or, think of an insurance company that has a major claims department. Claimants may present scores of claims daily. Some claims may be settled immediately; other may require recording, nomination of an assessor, assessment, recording of the assessment, scheduling an internal review, holding an internal review, recording of the internal review's minutes, correspondence with various stakeholders, scheduling of more internal reviews, holding more internal reviews, recording of these reviews' minutes, negotiation, even more internal reviews, nomination of a legal team, briefing of the legal team, scheduling even more internal reviews, litigation, drafting of cheques, signing the cheques and only then are the claims settled.

Surely departments with business processes as numerous and convoluted as this would benefit from an engine that would hold a detailed model of their business process on one hand, the work items on the other hand, and would allocate work items to workflow participants fairly. This engine falls under an approach named workflow management.

The second edition of the Oxford English Dictionary, accessed on line on June 2009, defines work as:

Definition §1.3
work, n. Something that is or was done; what a person does or did; an act, deed, proceeding, business; in pl. actions, doings [...] often collectively [25].

and workflow as:

Definition §1.4
work flow, n. in an office or industrial organization *[sic]*, the sequence of processes through which a piece of work passes from initiation to completion [25, accessed 19-March-2011].

The lexicographer was insightful. With the definitions of "work" and "work flow" on hand we can learn some of the important features that a workflow engine should have. A workflow engine shall:

(a) Record "the sequence of processes through which a unit of work passes from initiation to completion".

(b) Record the skills of the "person" or the "collective".

(c) Mark the "person" or the "collective" as available.

(d) Record "something" as "needs to be done".

(e) Classify that "something" as "an act, deed, proceeding, business, actions, doings".

(f) Fairly assign that "something" that "ought to be done" to a "person" or to a "collective" who are available, and are sufficiently skilled.

(g) Decide which parts of that "something" done by a "collective" can be done in parallel.

(h) Record that "something" "was done".

Indeed the protagonist in this dissertation is the workflow modelling language, the artifact used to record:

> "the sequence of processes through which a unit of work passes from initiation to completion" [25, accessed October 2010].

A workflow engine is not needed for the strategic management of big once off projects. They have dedicated project managers who record their sequence of activities in Gantt charts and risk registers. Workflow engine are needed for the management of those multitude work items that the organisation processes day in and day out. Unlike the project managers who use two lightly coupled artifacts — Gantt charts and risk registers, workflow configuration officers use a single document called "workflow model" to define all imaginable paths any work items may pass. The "workflow model" combines in a single comprehensive artifact information that the project manager records in two artifacts. Nonetheless, management of these large once off projects may benefit from the use of workflow engines for tactical activities such as the dispatching of tasks derived from the project plan to stakeholders, manage the approvals of leaves, enforce change control policies, coordinate procurement and more.

With the workflow model on hand, depending on the specific attributes of each work item, the workflow engine prescribes the specific activities that each workflow participant should perform as appropriate for each work item.

So that a workflow modelling language would be able to adequately describe a process, a workflow modelling language should:

- have high expressive power, namely be capable of expressing complex control patterns including dependency between activities, parallelism, and alternatives flows.
- be free of ambiguities,
- be readable,
- be maintainable, and
- have a formal mechanism to prove whether a workflow model is safe, namely if each workflow model is free of dead and live locks, will eventually terminate, and has no unreachable activities [3].

Many proprietary workflow modelling 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 model, and can be formalised with Petri nets or similar representations. For example, some vendors of industrial workflow engines, such as Software AG and TIBCO, deploy dialects of BPMN, a modern flow charting language, as their workflow modelling languages. 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 of each process in a special format, for the purpose of controlling the execution in the workflow engine [86].

1.3 Overview of the Research Program

Currently, organisations ask the workflow configuration officers to model the business processes in a special language. We suggest a different approach. We make use of a well accepted language for eliciting system requirements — the written use case language. Use cases are commonly produced during the requirements elicitation stages of projects and are thus available to the workflow configuration officers *naturaly*. We conjecture that these use cases are very readable, that they can contain most of the information needed to configure workflow engines and that all the workflow configuration officer has to do is to extend the use cases as we describe below.

Here is a brief overview of our approach from the point of view of its users; much more detail is given below. With a use case set on hand, the workflow configuration officer creates a business transaction routing sheet from each scenario, each describing several activities which should be performed as a group. When a work item arrives to the organisation, the first workflow participant to touch the work item catalogues the work item, and records its attributes. Using these attributes the workflow engine links routing sheets to the work item. Following this, workflow participants perform activities according to the routing sheets, until eventually an activity is found that this workflow participant should not deal with, at which point the workflow engine passes the work item to another workflow participant. As the workflow participants execute each activity, the workflow participants acknowledge this to the workflow engine. From time to time the workflow engine records audit data describing the workitems' attributes and progress.

Before the use case oriented workflow engine approach would be widely accepted, one would need to see if the approach is better (or at least as good) as the incumbent approach from several perspectives — feasibility, richness of expressive power, readability, maintainability and safety. In this dissertation we report on research that explores the extent to which this is so.

We translated these perspectives into hypothesis and conjecture that arifact employing a use cases language are readable and contain most of the

information needed to configure workflow engines and that all the workflow configuration officer has to do is to extend the use case as we describe in section 3.4 on page 72 — Use Case Oriented Workflow Engine.

The hypothesis that we present, test and assess in this dissertation are:

H_1: The written use case language can be used as a workflow modelling language.

H_2: The written use case language can express *common* workflow patterns.

H_3: The written use case language can express every *reasonable* workflow pattern.

H_4: The written use case language can be tested for safety.

H_5: The readability of the written use case language is higher than the readability of incumbent languages.

By testing the hypothesis we lay the theoretical foundations for the workflow industry to improve the readability and maintainability of its products, which in turn could simplify the role of workflow configuration officers and increase the capability of workflow engines to handle complexity. In particular, we test H_2 by exemplification, using a library of 43 patterns. We then test the much wider H_3, as it includes the adjective "every", mathematically using tools provided by the Petri net community.

We start at Chapter 2 by presenting a description of the state of the art, drawing on the achievements of three communities: business process management, requirements engineering and human computer interaction. As our primary concern is language we also draw upon work from the linguistics and the Literacy communities.

In Chapter 3 — Architecture, we provide a logical design of a use case oriented workflow engine. The design is detailed enough to enable a software developer to write a physical design of use case oriented workflow engine. It is also sufficiently detailed for a workflow configuration officer to learn how to configure a use case oriented workflow engine, for a workflow participant to learn how to operate a use case oriented workflow engine, and for a business process manager to learn how to analyse the logs produced by the business activity monitor, a building block of a use case oriented workflow engine.

In Chapter 4 — Case Study, we describe in detail the implementation of a use case oriented workflow engine in BT Financial Group, the financial

services arm of Westpac Banking Corporation. By doing so we test H_1. In Section 5.2 — Patterns Approach, we demonstrate that *common* workflow patterns can be expressed using the written use case language, thus testing H_2. In Section 5.3 — Sound Workflow Nets Approach, we further generalise our investigation and provide a proof that *every reasonable* workflow pattern can be expressed using the written use case language, thus we test H_3. Then in section 5.4 — Unsafe Patterns, we provide three examples of use cases that are not safe, and call upon existing work to demonstrate how the safety violation can be identified; this tests H_4. In Chapter 6 — Readability, we assess the readability of the written use case language, showing H_5 and completing an evaluation of the hypotheses above.

Chapter 2

Context

> Rabbi Hanina said: 'He who attributes his quotes, brings salvation to the world.'
>
> [36, 4:4]

HAVING EARLIER SUGGESTED that a use case language may be used as a workflow modelling language, we now establish the lineage of our approach. We draw upon work drawn from three computer science research communities: the business process management community within Databases and Information Systems field, the requirements engineering community within Software Engineering, and the human computer interaction community. As our primary concern is languages we also draw upon work from the linguistics and the literacy communities.

2.1 Business Process Management

The business process management community is mostly concerned with the round trip of modelling, verification, enactment, monitoring, and tuning of business processes within the organisation and between organisations. A wider view also encompasses organisational structure, leadership and performance management, as well as service oriented architecture (SOA) [7].

Amongst the assets of the community are an annual conference — BPM, YAWL — a Workflow Language [4], a pattern library [96], a seminal paper on Petri nets by Tadao Murata [79], a business process management adoption of Petri nets by van der Aalst [2] and Workflow terminology and glossary [117] which was written in collaboration with the industry. In this section we focus on the modelling, verification, enactment, monitoring, and tuning of business processes within the organisation and between the organisations.

2.1.1 Modelling

Key to a productive discussion about business processes is an accurate, readable and verifiable model of the business process. Here we discuss several approaches to the modelling of workflow; we then see how does the BPM community assesses the readability of the models empirically and how does the community verify that the models are safe.

Workflow Modelling Languages

Mili *et al* sorted through the workflow modelling languages in a survey entitled, alphabet soup, describing 15 languages [74]. Adopting the Workflow Management Coalition's terminology [117], Mili *et al* articulated A first-cut business process meta model. The meta-model is implicitly broken into two domains: organisation and process. An organisation has a hierarchy of the departments within it, each department being an organisation in its own right. An organisation has many members or actors, each capable of fulfilling one or many roles. The organisation performs many functions, calling upon the support of role bearers. A process is composed of sub-processes, each being a process on its own right. At a certain point the business process modeller decides that a sub-process is sufficiently granular so that it can be broken into one or many activities. An activity consumes resources and produces resources. An activity is triggered by an event and may trigger other events. Mili *et al* bridge the organisational domain with the process domain by observing that role bearers perform activities.

Mili *et al* classify workflow modelling languages according to the tradition that derived them: MIS (the IDEF family of languages and Petri net), Workflow and process integration (RosettaNet, ebXML and BPEL4WS to name just a few), and object oriented (for example the UML Activity diagram). Mili *et al* state, though, that UML does not yet provide explicit support for business process modelling [74, Page 47]. Another dimension of Mili *et al's* taxonomy is target audience who may be workflow participants, business analysts or machines (*e.g.* the XML style languages such as ebXML is aimed at the latter).

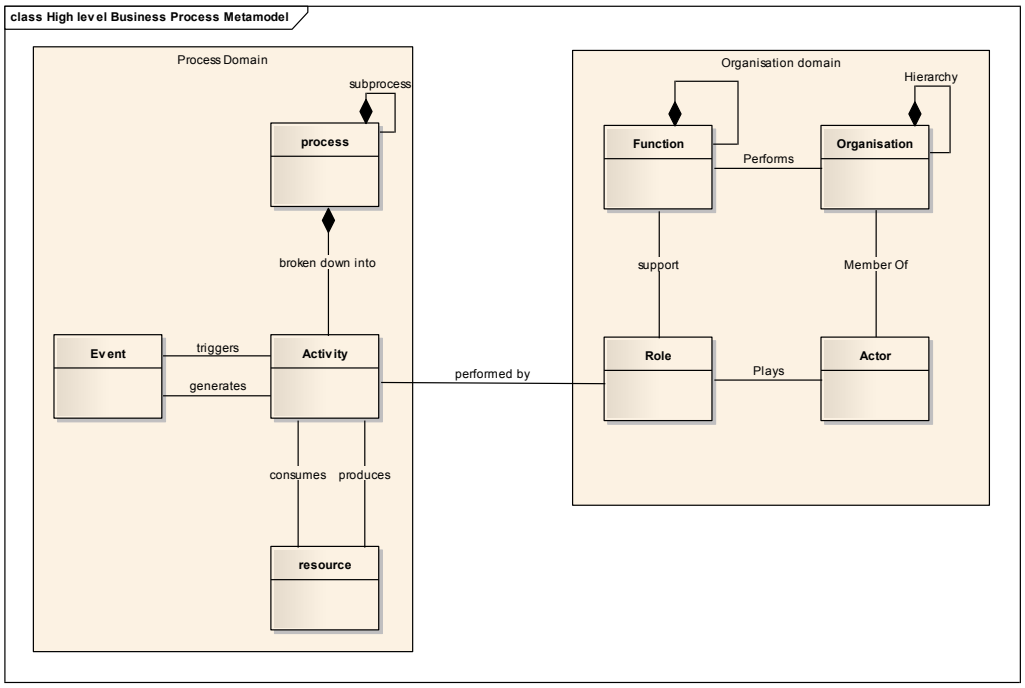


Figure 2.1 – Mili *et al's* first-cut business process meta model [74]

BPMN

We now present three of the many languages proposed for the modelling workflow: BPMN which is designed with business analysts in mind [112], the UML Activity Diagram [19, pages 270-277] which is designed with objects in mind [97], and Petri nets, a formal language with mathematical foundations that can be used to model workflow [79].

The Business Process Modelling Notation (BPMN) was developed by the Business Process Management Initiative (BPMI), and is currently maintained by the Object Management Group since the two organizations merged in 2005. On December 2010, the version of BPMN was 1.2, with a major revision process for BPMN 2.0 in Beta 2 stage. BPMN depicts the end to end flow of a business process. The language has been specifically designed to coordinate the sequence of processes and the messages that flow between different process participants in a related set of activities. A sample diagram employing the BPMN is provided in figure 2.2 on page 16.

IBM Corporation's Stephen A. White succinctly described the BPMN language, we quote directly from his 2004 paper:

> **Definition §2.1**
> BPMN, n. A modelling language that is readily understandable by all business users, from the business analysts who create the initial drafts of the processes, to the technical developers responsible for implementing the technology that will perform those processes, and, finally, to the business people who will manage and monitor those processes [112].

BPMN models a business process using a diagram, which is based on a flowcharting technique tailored for creating graphical models of business process operations. A Business Process Model, then, is a network of graphical objects, which are activities (i.e., work) and the flow controls that model their order of performance. With BPMN, a business process diagram is made up of a set of graphical building blocks. The building blocks were chosen to be distinguishable from each other and to utilise shapes that are familiar to most modelers. For example, activities are rectangles, and decisions are diamonds. This provides a small set of language categories so that the reader of a business process diagram can easily recognise the basic types of building blocks and understand the diagram. Within the basic categories of building blocks, additional variation and information can be added to support the requirements for complexity without dramatically changing the basic look and feel of the diagram. Wohed *et al*, with the help of a workflow

patterns library (more about it in section 2.1.1 — Evaluating Languages), found BPMN suitable for workflow modelling with some reservations [115].

The Unified Modelling Glslanguage (UML) takes an object oriented approach to the modelling of applications, while BPMN takes a process oriented approach to modelling of systems. Where BPMN has a focus on business processes, UML has a focus on software design and therefore the two are not competing languages but are articulating different views on systems. The BPMN and the UML are compatible with each other. A business process model does not necessarily have to be implemented as an automated business process in a process execution language. Where this is the case, business processes and participants can be mapped to constructs such as use cases and behavioural models in the UML. Russell *et al*, with the help of a workflow patterns library, found the UML Activity Diagram suitable for workflow modelling with some reservations [97].

Petri Nets

Invented in August 1939, at the age of 13, by Carl Adam Petri to describe chemical processes [11], Petri nets are appropriate for the description of many varieties of system, including reactive control, operating systems, and business software. Murata's description of the language is very succinct:

> Petri net is particular kind of directed graph, together with an initial state called the initial marking, M_O. The underlying graph N of a Petri net is a directed, weighted, bipartite graph consisting of two kinds of nodes, called places and transitions, where arcs are either from a place to a transition or from a transition to a place. In graphical representation, places are drawn as circles, transitions as bars or boxes. Arcs are labelled with their weights (positive integers), where a k weighted arc can be interpreted as the set of k parallel arcs. Labels for unity weight are usually omitted. A marking (state) assigns to each place a non negative integer. If a marking assigns to place p a non negative integer k, we say that p is marked with k tokens. Pictorially, we place k black dots (tokens) in place p [79].

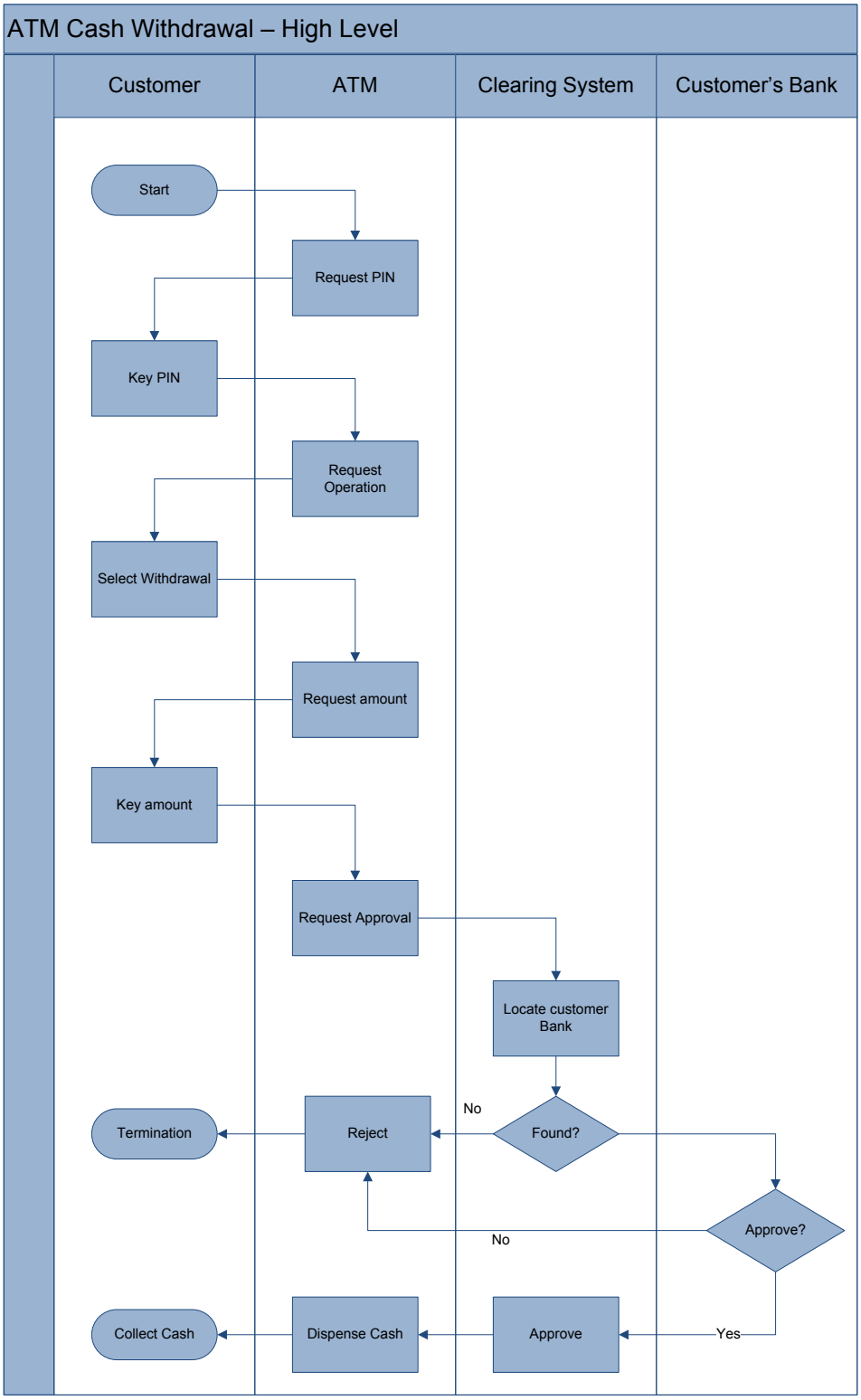


Figure 2.2 – Basic BPMN example. The example is of equivalent content to the written use case language example in figure 2.7.

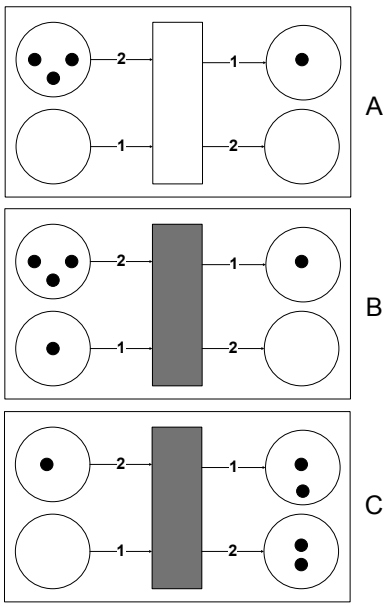


Figure 2.3 – Petri net example. Here we show the firing sequence of a Petri net. In state A the transition is not triggered because the place in the bottom left does not have a token. In state B the transition is triggered with a token moving from the bottom left place to the bottom right place. In state C, after firing, each input place loses an equal number of tokens to the weight of the arc flowing from it, and each output place gains an equal amount of tokens to the weight of the arc pointing at it [74, Page 17].

Please refer to figure - 2.3 on the preceding page for an example. Formally a Petri net is defined as:

Definition §2.2
Petri net, n. A triple (P, T, F) where P is a finite set of places, T is a finite set of transition ($P \cap T = \phi$), and $F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs (flow relation) [3].

To understand what behaviour is encoded in a Petri net, it is common to use a simulator [22]. Simulation can also by applied, as we describe later, to assess properties of a Petri net. The discussion about workflow should be conducted formally [7]. We call upon van der Aalst and list some building blocks of the grammar of the Petri net algebra [3].

1. A place p is called an *input place* of a transition t iff there exists a direct arc from p to t.

2. A place p is called an *output place* of a transition t iff there exists a direct arc from t to p.

3. $\bullet t$ denotes the set of input places of transition t.

4. $p\bullet$ is the set of transition sharing p as an input place.

5. A state represented as $1P_1 + 2P_2 + 1P_3 + 0P_4$ is the state with one token in place P_1 two tokens in place P_2, one token in place P_3 and no tokens in place P_4. Or in short $P_1 + 2P_2 + P_3$

6. To compare two states M_1 and M_2 one can say that $M_1 \leq M_2$ iff for all $p \in P : M_1(p) \leq M_2(p)$.

7. $M_1 \xrightarrow{t} M_2$ denotes that transition t is enabled in state M_1 and that firing t in M_1 result in state M_2.

8. $M_1 \rightarrow M_2$ denotes that there is a transition t such that $M_1 \xrightarrow{t} M_2$.

9. $M_1 \xrightarrow{\sigma} M_2$ denotes that the firing $\sigma = t_1 t_2 t_3 ... t_{n-1}$ leads from M_1 via set of intermediate states $M_2 ... M_{n-1}$ to state M_n.

10. $M_1 \xrightarrow{*} M_2$ denotes that a state M_n is *reachable* from state M_1 (that is, there is $M_1 \xrightarrow{\sigma} M_2$).

11. $P(PN, M)$ denotes a Petri net PN with initial state M.

Petri nets can be used to specify the routing of work items. Activities are modelled by transitions and causal dependencies are modelled by places and arcs. A place corresponds to a condition which can be used as pre and/or post condition for activities. An AND split corresponds to a transition with two or more output places, and an AND join corresponds to a transition with two or more input places. "OR splits" and "OR joins" correspond to places with multiple outgoing or ingoing arcs [3].

Definition §2.3
workflow net, n. A Petri net that has one input place i and one output place o, and where for each transition t there is a path from i to o via t. Formally a Petri net $PN = P(P, T, E)$ is a workflow net *iff* (i) there is one source place $i \in P$ such that $\bullet i = \phi$, there is one sink place $o \in P$ such that $\bullet o = \phi$, and (ii) every node $x \in P \cup T$ is on the path from i to o [3].

Workflow nets are classical Petri nets without data, hierarchy, time and other extensions, therefore, their convenience is limited [4]. A sound workflow net is a workflow net further constrained so that the markings are bounded. During execution of a sound workflow net the number of tokens remains limited. A sound workflow net is safe *iff* for any case, the process terminates properly, i.e., termination is guaranteed, there are no dangling references, and deadlock and livelock are absent. Formally:

Definition §2.4
sound workflow net, n. Workflow net $PN = (P, T, F)$ is sound *iff* (i) for every state M reachable from state i, there exists a firing sequence leading from state M to state o $(\forall_M (i \xrightarrow{*} M) \Rightarrow (M \xrightarrow{*} 0))$, and (ii) state o is the only state reachable from state i with at least one token in place o $(\forall_M (i \xrightarrow{*} M \wedge M \geq o) \Rightarrow (M = o))$, and (iii) there are no dead transitions in (PN, i) ($\forall_{t \in T} \exists_{M,M'} i \xrightarrow{*} M \xrightarrow{t} M'$) [3].

Definition §2.5
reachability graph of a net, n. A 5 tuple (V, E, T, L, v_O), where (V, E) is a finite, labeled directed graph with the property that for each v in V , there exists a directed path from v_0 (called the initial vertex) to v, *i.e.*, (V, E) is rooted at v_0. Each (directed) edge in E is labeled by an element in the transition set T (of a Petri net) [84].

Mayr has demonstrated an algorithm for the general Petri net reachability problem [70].

Evaluating Languages

Members of the Business Process Management community evaluate these languages using the following criteria:

Readability To ensure proper review of models, business users should read and understand process models, whether written or drawn. We describe the mechanics of reading in section 2.5 — Literacy. Process models are the building blocks of the human computer interface (HCI). We describe the HCI community's work in section 2.3 — Computer Human Interaction.

Maintainability As we demonstrate below using activity theory in section 2.1.5 — Tuning, business process definitions stale fast and constant maintenance is required. Authorised users of workflow engines should be able to alter them.

Mathematical foundation With 70 years [11] of work devoted into Petri net research, the community established a well defined algebra. The community leverages its investment by translating between other language and Petri nets [6, 22, 32, 70, 52, 79, 84]. Indeed Lee *et al* show that use cases can be transformed into Petri nets [66].

Expressive Power By the expressive power of a language, we understand the set of all queries expressible in that language [29].

The idea of a catalogue of patterns originated in architecture in the work of Alexander [16]. It became widely accepted in the software industry following the publication in 1995 of the "Gang of Four" book listing patterns of object oriented design [44]. In the business process modelling domain, a seminal work has been done by the Workflow Patterns Initiative [96] which started in 1999 and 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 aim of this initiative is to provide a conceptual basis for process technology. In doing so, they established a common vocabulary and agenda often used when discussing workflow subjects.

The Workflow Patterns Initiative identified 43 patterns, that seem to appear often in describing business processes. While definitely not exhaustive, the library is rich enough and is thus used as a benchmark for expressive power [5, 96, 97, 115]. The patterns are named WPC 1 to WPC 43. Some patterns are presented in Petri net language, in a fairly abstracted format (with steps labelled A, B, C etc); sometimes they are also illustrated with an example where the action steps have meaningful names.

Because the workflow pattern library is extensive and richly descriptive, it provides a good test bed for new approaches to business process modelling. This approach has been taken by Russell *et al* [96], when they evaluated the richness of UML2.0 activity diagrams [97] and by Wohed *et al* [115], when they evaluated the richness of BPMN.

Evaluating Readability of Modelling Languages

Researchers use many criteria to evaluate language, including expressive power [97], precision which is essential to support formal analysis [65], terseness [107], aesthetics [111], and usability [106]. Nielsen suggests that usability is about ease of learning, efficiency, memorability, errors, and satisfaction [81, Page 25]. Thus one aspect of usability of a language concerns the ease with which writers can express their ideas, a second concerns readability, a third concerns the ease of learning the language. There may be a tradeoff between the three aspects; among these we focus in this section on readability, an aspect of usability that lends itself to empirical examination.

Definition §2.6

readability, n. a language's success in accurately communicating an idea from the writer to *human* readers [as oppose to — from the writer to *computer*] [83].

Siau *et al* [103] presented a three dimensional taxonomy for the evaluation methods of languages: feature comparison, theoretical and conceptual evaluation, and empirical evaluation. The empirical dimension was further segmented into: surveys, laboratory experiments, field experiments, and case studies.

Gemino *et al* [45] further elaborated Siau *et al*'s taxonomy for empirical studies. Among other contributions, they characterised some dimensions of the taxonomy of laboratory experiments, namely: the type of instruments presented to participants, the procedure participants are asked to complete and the element measured.

Gemino *et al* observed two types of instruments presented to participants, between-grammar instruments and within-grammar instruments. When conducting within-grammar experiments researchers vary the writing style of artifacts (*e.g.* [100]), or, vary the training given to participants (*e.g.* [27]).

In [100], Si *et al* created sample corpus of 91 Web documents with variable sentence length distribution and three readability levels. When using randomly collected web pages Si *et al* arrived at mixed results, in part because of the small amount of training data and the large amount of variation in the training data. Si *et al* then adopted a second approach to acquiring data — use the syllabi of elementary and middle school science courses. Three sets of syllabi (one per readability level) were collected from different Web sites. The experiments showed that the a readability model that contains both surface linguistic and content-based features is much more accurate on K-8 science Web pages than the widely used Flesch-Kincaid [61] readability metric.

In [27], Cox *et al* started from the premises that use cases cases rely predominantly upon natural language and that for this reason, research groups have proposed guidelines to assist in writing use cases. Various research groups have found that writing guidelines help. However, Cox *et*

al experience with students was that some guidelines were a little unwieldy, and were difficult to apply. Consequently Cox *et al* propose some simplified use case guidelines. Cox *et al* then conducted an experiment to explore whether the simplifications result in any loss of use case quality, and found that the simpler guidelines were as effective as the more complex guidelines.

Gemino *et al* observed two types of activities: the reading of instruments (*e.g.* [15, Pages 64-74]), and the writing of instruments (*e.g.* [83]).

In [15, Pages 64-74] Tullis *et al* present a methodology for comparing the usability of artifacts by measuring task success. Tullis *et al* emphasise that defining task success is at times not simple and that criteria should be established such as asking the participants to exercise as much effort as the participant would exercise outside the laboratory, apply the "three strikes and you are out rule", which means that participants will be allowed to make three attempts before the experimenter stops them, or "Call" the task after a specified time.

In [83] Norman is challenged to measure the match of psychological variables of interest to the physical variables being controlled. For example, when using a tap water a physical variable is the ratio of hot and cold water and the interest is the temperature of the water. In a typical experiment Norman would, for example, compare the usability of single tap mechanism with the two taps system.

Gemino *et al* observed that the measured element of the experiments were: correctness, efficiency (that is, time to complete a task using the language, *e.g.* [14]) or the process participants follow (*e.g.* [60]).

In [14], Aguirre-Urreta *et al* review a dozen studies comparing the usability of artifact employing an entity relation languages with artifacts employing object oriented languages. The main tenet of their review is that researchers appear to have approached empirical comparison using a 'black-box' approach: given a controlled input, the focus of analysis has been on comparing alternative output without consideration for the particularities of the process that mediates between them. Aguirre-Urreta *et al* suggest that languages should be designed to account for the cognitive architecture of their users, as opposed to being tied to a certain pattern library.

In [60], Kim *et al* started from the premises that the cognitive process used when reviewing an artifact has two facades: perceptual and conceptual. The perceptual process is a bottom-up activity of sensing something and knowing its meaning and value, while the conceptual process is a top-down activity of generating and refining hypotheses. In other words, we search and recognise relevant information through perceptual processes and reason by inferring and deriving new information through conceptual processes. To compare two languages, they articulated a problem drawn from the fast food industry. Rather than measuring the accuracy of the participants (all students), the participants were trained to "think-aloud". The subjects were then presented with the experimental diagrams and were asked to diagnose the business system based on the diagrams. Kim *et al* used protocol analysis to investigate the cognitive process involved in diagnosing the business processes through the use of multiple diagrams.

When measuring effectiveness, Gemino *et al* distinguish between "comprehension", which refers to what the reader can answer about particular elements of the language, and "domain understanding", which is shown by problem-solving questions requiring significant additional cognitive processing. To compare the effectiveness of artifacts in the usual way, researchers ask participants to read an instrument and then measure participants' answers to questions about the domain described in the instrument. A confounding aspect for this measurement might be the different levels of initial domain knowledge participants bring with them.

Definition §2.7

initial domain knowledge, n. The original familiarity of participants with the material presented in the artefact.

Researchers have sometimes implicitly assumed homogeneity of initial domain knowledge within community by choice of a well-known domain (*e.g.* restaurant [60], ballistic trajectory [48], elevator [50]) or else they generate synthetic artifacts, thus removing any domain specific information from the artifacts (*e.g.* [72, 111]). We have not witnessed an experiment comparing modelling languages that measures initial domain knowledge explicitly. In

contrast, in the medical area it is a common practise to compare a treatment against the use of a *placebo*.

Definition §2.8
placebo, n. A substance with no therapeutic effect used as a control in testing new drugs [25, Accessed 2-June-2010].

For further control, medical researchers randomly assign subjects to either the control group (who are treated with a placebo) or the experimental group. The experimenter is not told which treatment the experimenter administers to each participant. This type of experiment is called *randomised, double blind, placebo controlled*.

Wohlin [116, pages 43, 49] characterised the context or setting of empirical work dealing with software engineering, along several aspects. There can be one or multiple subjects, who work with one or multiple objects (software artifacts). The experiments may be on-line (within real projects) or off-line; the subjects may be students or professional practitioners; the objects may come from real projects, or be toy examples; and the study may be specific to the context, or aimed at conclusions of general validity across the software engineering domain. We will return in Chapter 3 — Architecture to the modelling of business process. There we will discuss the proposition that one can call upon written use case languages to model business processes.

2.1.2 Verification

We will now learn how the BPM community verifies workflows. The community has the tools needed to prove that a workflow model is safe, namely (*i*) that each workflow model is free of dead locks, (*ii*) that each workflow model is free of live locks, (*iiii*) that each workflow model will eventually terminate, and (*iv*) that each workflow model has no unreachable activities [3].

In [3] van der Aalst suggests three different ways to check if a workflow net is sound: (*i*) apply brute force in polynomial time, (*ii*) verify that some suspicious constructs are absent, or (*iii*) partition the workflow net into

sound sub workflow nets and apply brute force in polynomial time to each of the sub workflow nets. Here are core points from his paper:

> **Definition §2.9**
> live Petri net, n. A Petri net is live *iff*, for every reachable state M' and every transition t there is a state M'' reachable from M' which enables t.

> **Definition §2.10**
> bounded, a. A Petri net is bounded *iff* for each place p there is a natural number n such that for every reachable state the number of tokens in p is less than n.

> **Definition §2.11**
> safe, a. A Petri net is safe *iff* for each place the maximum number of tokens does not exceed 1.

Safety is a desirable property, because it makes no sense to have multiple tokens in a place representing a condition. A condition is either true (one token) or false (no tokens).

> **Definition §2.12**
> structurally bounded, a. A Petri net is structurally bounded if the net is bounded for any initial state.

> **Definition §2.13**
> well formed, a. A Petri net PN is well formed *iff* there is a state M such that (PN, M) is live and bounded.

Paths connect nodes by a sequence of arcs. A path C from a node n_1 to a node n_k is a sequence $(n_1, ..., n_k)$ such that $(n_i, n_{i+1}) \in F$ for $1 \leqslant i \leqslant K-1$. C is elementary iff, for any two nodes n_i and n_j on C, $i \neq j \Rightarrow n_i \neq n_j$. C is conflict free for any place n_j on C and any transition n_i on C, $j/neqi-1 \Rightarrow n_i \notin \bullet n_i$. For convenience van der Aalst introduce the alphabet operator α on paths. If C $(n_1, n_2, ..., n_k)$, then $\alpha(C) = \{\ n_1, n_2, ..., n_k\ \}$.

A Petri net is is strongly connected *iff*, for every pair of nodes (*i.e.*, places and transitions) x and y, there is a path leading from x to y. A Petri net is a free choice Petri net *iff*, for every two transitions t_1 and t_2, $\bullet t_1 \cap \bullet t_2 \neq \phi$ implies $\bullet t_1 = \bullet t_2$. Given a WF net $PN = (P, T, F)$, one wants to decide whether PN is a sound workflow net. In [1] van der Aalst shows that soundness corresponds to liveliness and boundedness by defining an extended net $\overline{PN} = (\overline{P}, \overline{T}, \overline{F})$. $\overline{PN}$ is a Petri net obtained by adding an extra transition t^* which connects o and i.

Formally, the extended Petri net $\overline{PN} = (\overline{P}, \overline{T}, \overline{F})$ is defined as follows:

1. $\overline{P} = P, \overline{T} = T \cup \{t^*\}$ and
2. $\overline{F} = F \cup \{\langle o, t^* \rangle, \langle t^*, i \rangle\}$ and
3. $F \cup \{\langle o, t^* \rangle, \langle t^*, i \rangle\}$.

In [1] van der Aalst proves that a WF net PN is sound *iff* the $(\overline{PN}, i)$ is a live Petri net and bounded thus showing that standard Petri net based analysis techniques can be used to verify soundness. However, for a complex WF net using this technique may be very slow. Indeed a general solution to the problem is TBA. Thus the community narrows the discussion to free choice workflow nets of which to decide if they are sound can be done in polynomial time. Furthermore, van der Aalst shows in [3] that a sound free choice Workflow net is safe.

Fahland *et al* evaluated the soundness of 735 industrial business process models in a few milliseconds per process using a sequence of checks in IBM WebSphere Business Modeler, LoLa and Woflan. These few milliseconds Fahland *et al* included the compilation of the models into Petri net [37].

Another approach to obtain a structural characterisation of 'good' workflows, is to balance AND/OR splits with respective AND/OR joins. Clearly, two parallel flows initiated by an AND split, should not be joined by an OR join. Two alternative flows created via an OR split, should not be synchronized by an AND join. A workflow configured this way is called *well handled*, and formally: Petri net PN is well-handled *iff*, for any pair of nodes x and y such that one of the nodes is a place and the other a transition and for any pair of elementary paths C_1 and C_2 leading from x to y, $\alpha(C_1 \cap \alpha(C_2) = \{x, y\} \Rightarrow C_1 = C_2$.

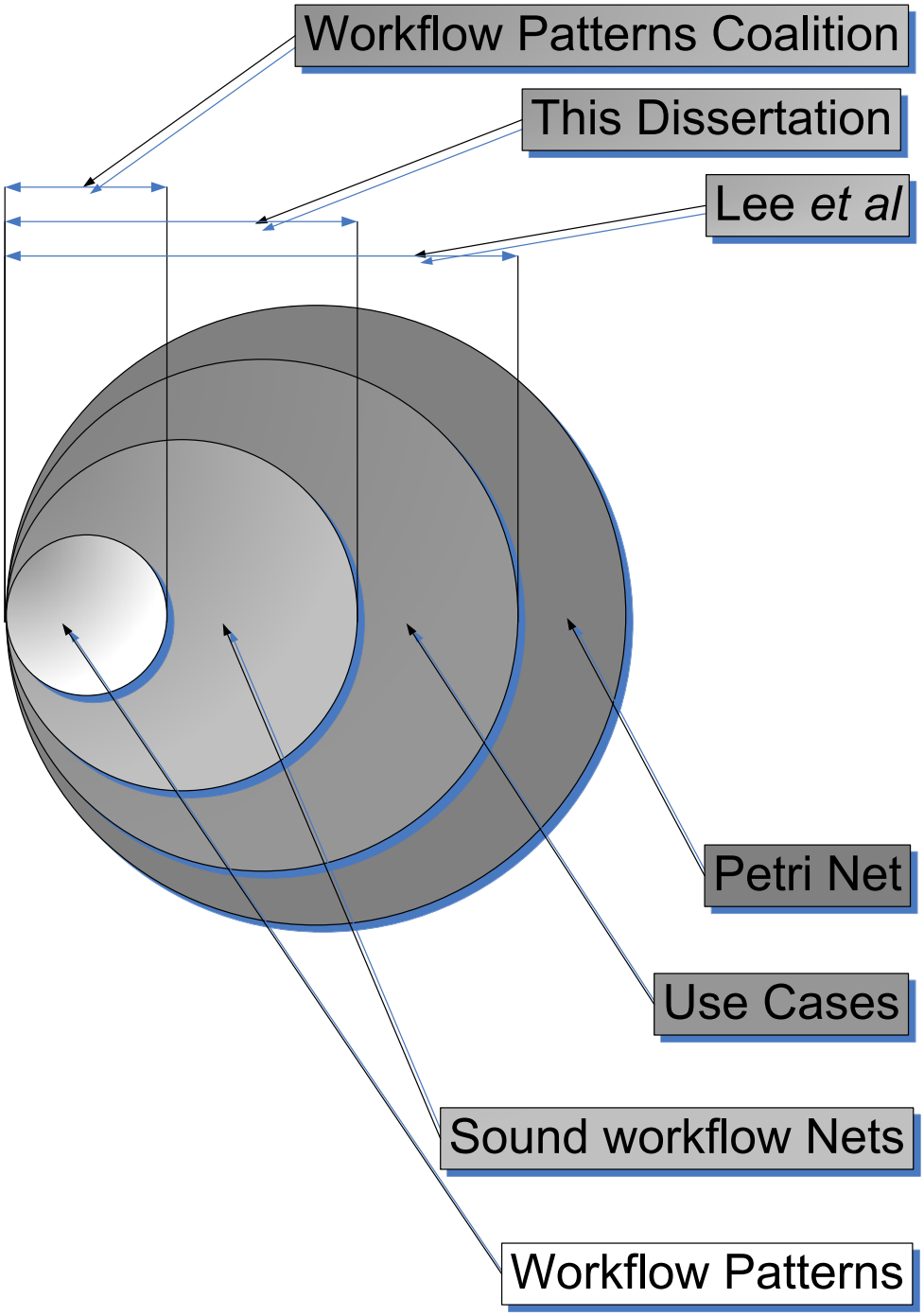


Figure 2.4 – Classification

2.1.3 Enactment

A workflow can be modelled and presented to workflow participants as such, but to ensure compliance with the model some organisation may find it beneficial to use programmes that enact the workflow models — workflow engines. Here we define the building blocks of a workflow engine. In particular we highlight the integral nature of a workflow modelling language and describe some building blocks that any workflow modelling language should incorporate.

In early times, transaction processing systems were designed to support a small portion of the business processes – the recording of the business transactions. Administrators invoked those applications once all required data 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 queues and processed work sequentially. In some cases the paradigm used to describe these queues is that of trays in an office [7].

Today, it is common for some business processes to be processed over several session as data drips into the organisation(s) and for the processing of business transactions to span over multiple organisations, systems and organisational role bearers. For this type of work a dedicated class of software products emerged – workflow engines. The Workflow Management Coalition, an industry body, uses the compound word ‘‘workflow” for the previously coined term *“work flow”* and define workflow as follows:

> **Definition §2.14**
> workflow, n. 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 [117, Page 8].

van der Aalst *et al* in turn defines workflow engine as follows:

> **Definition §2.15**
> workflow engine, n. A generic software system driven by explicit process design to enact and manage operational business process [95].

A workflow model is an explicitly required element of a workflow engine. Without an explicit process modelling languages, a software program is not a workflow engine, even if the software program does manage some aspects of workflow such as routing. E.g. Microsoft Dynamics CRM 4.0 has some workflow capability, but has no explicit workflow modelling language, and is thus not a workflow engine.

Definition §2.16
workflow modelling language, n. A format for defining a process enacted by a workflow engine.

The atomic building blocks of a workflow model are activities and roles which are instantiated at run time as work items and workflow participants.

Definition §2.17
activity, n. A description of a piece of work that forms one logical step within a process. An activity may be a manual activity, which does not support computer automation, or a workflow (automated) activity. A workflow activity requires human and/or machine resources(s) to support process execution; where human resource is required an activity is allocated to a workflow participant [117, Page 13].

An activity is the *atomic* piece of work dispatched by the workflow engine to workflow participants. If an organisation charges for the performance of activities on time and material bases, and the execution of an activity is long, an activity can be further broken into time sheet sessions (think of an engineer to whom the workflow engine dispatches a complex design activity that lasts five eight hours business days, with breaks for meetings, and meals in the middle).

Definition §2.18
workflow participant, n. A resource which performs the work represented by an activity instance [117, Page 18].

A way to describe workflow participants is by assigning them roles.

Definition §2.19
role, n. A group of workflow participants exhibiting a specific set of attributes, qualifications and/or skills [117, Page 53].

Each activity is associated with one and only one role. A role can be associated with many activities (a job description can be complex). A workflow participant may have many roles (wear many hats).

Definition §2.20
work item, n. The representation of the work to be processed by a workflow participant in the context of an activity within a process instance [117, Page 19]..

A compound unit of management in the workflow domain is a worklist.

Definition §2.21
worklist, n. A set of work items associated with a given workflow participant [117, Page 20].

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.

2.1.4 Monitoring

An organisation that deploys a workflow engine would expect that the engine, in addition to the allocation of work items to workflow participants, would record the history of the work items. Management uses this log to support process improvement drives.

Definition §2.22
business activity monitor, n. A utility that provides the ability to monitor track and report on workflow events during [and after] workflow execution [117, Page 56].

Modelling a process is a complicated and time consuming activity. Modelling requires deep understanding of the process at hand (*ie.* long discussions with workers and management). Typically there are discrepancies

between the actual workflow process and the process perceived by various stakeholders. Often, modelling states what should be done rather then describe the actual process. Process modelling is done *a priori*. Business process monitoring is done *post priori* and can thus agree or conflict with the business processes model. The challenge of the business process practitioner is to reverse engineer the process, identify the paths commonly trailed and influence the process so that the desirable paths are trailed more often. For example, business process management may what to analyse how often certain exception handling processes are invoked.

Business process monitoring does not necessarily assume the presence of a workflow engine. Enterprise resource planning systems (ERP) or customers relationship management systems (CRM) generate these event logs as well. Naturally, the more control the business process practitioner has over the format of the event log, the easier it is to derive significant conclusions from the event log. Ideal event logs list the activities performed on work items in correct order. It is important to state that monitoring is not a redesign activity. The goal is to understand the process first, and only then improve the process. The goal is to generate an explicit representation of a process model [8]. Such event logs can be obtained from workflow engines, but in applications where there is limited support from process aware systems it may become difficult to retrieve log data in that format [39].

The challenges facings the process engineer are that not all possible paths are executed, that noise can obscure important issues, and that failure to carefully communicate logging of information about individual workflow participants may result in industrial disputes.

2.1.5 Tuning

Business process tuning is the step that closes the loop of round trip business process management, meaning the iterative process cycle from modelling, to verification, enactment, to monitoring, and to tuning is done in one integrated environment. Hardly any vendor or analyst explains *how* a BPM system delivers better tuning capabilities. It is almost considered to be common sense, like you don't have to explain why you get warm in the sun. But it is not that obvious at all [55].

Activity theory provides the theoretic foundation for dynamic business process tuning. It is a meta theory with roots in the German philosophy (from Kant to Hegel), in the writing of Marx and Engels, and in the Soviet cultural historical psychology of Lev Semyonovich Vygotsky (1896–1934), Alexander Romanovich Luria (1902–77) and Alexei Nikolaevich Leont'ev (1903–79) [80, Page 19]. Kuutti applied activity theory principles to the human computer interaction research [63]. Bradram applied the theory to business process management [21]. Adams *et al* described how Staffware, a workflow engine, applies activity theory principles using "event nodes" [12].

Proponents of the activity theory suggest that individuals, even animals, map activities that have beneficial rewards into a web of actions, which on their own have no meaning. Over time these maps become a tool accessible to other individuals and thus part of the social interaction [41]. In the modern BPM domain, we refer to "actions" as "activities" [117, Page 13] and we refer to "activities" as "business processes". Adams *et al* referred to "activities" as "worklets" [12]. So, using our terminology "Activity Theory" should have been called "Business Process Theory". For the remainder of this section we use current BPM terminology.

Workflow configuration officers formally construct process models from activities. As workflow configuration officers rely on their experience when constructing process models, they risk becoming detached from the working life.

These process models guide workflow participants toward an expected result. Often, workflow participants use the process model as a guide while enacting a business process. Empowered workflow participants synthesise activities from the process models and from the conditions of the concrete situation. This synthesis forms a feedback loop in the course of a business process and becomes the basis for learning, which is embedded in each business process. This learning process, in turn, extends the process models. Deviation from a process model is a breakdown, and therefore a potential learning situation. Breakdown situations are all too common; they are a natural and very important part of any business process. Deviations should form the basis for learning and thus for developing and extending process models.

It is important to consider exactly who is allowed to read, alter and save process models within a work practice, but this is a question of division of work and corresponding access rights within the workflow engine — not a separation of the process modelling and execution of work.

In order for process models to become resources for the future realisation of an activity, the review of the process model should be made as part of this activity. That calls for very readable workflow modelling languages. This understanding of process models as central dynamic assets of the organisation raises some unconventional requirements from workflow engines — instead of rigidly supporting routing information around the organisations, the workflow engine should mediate the anticipation of the workflow configuration officer with recurrent events in working life. Hence, a workflow engine should support the modelling, verification, enactment, monitoring, and tuning of process models within a democratic work activities.

2.2 Requirements Engineering

The Requirements Engineering community is concerned with the articulation of requirements, the management of large number of requirements, cataloguing of requirements and the cross referencing of requirements with software artifacts, as well as the automatic generation of software from formally articulated requirements. Its has an annual conference is RE. The latest, RE'10 was conducted in October 2010 in Sydney, Australia.

Requirements engineering is the branch of software engineering concerned with the real-world goals for, functions of, and constraints on software systems. It is also concerned with the relationship of these factors to precise specifications of software behaviour and to their evolution over time and across software families [119]. A more recent development in the field of requirements engineering is the introduction of use cases for elicitation of business requirement [18]. We ourselves adopt this language for the modelling of business process [86].

We now look at use cases which are a popular requirement elicitation tool. We define what is a use case, describe the UML use case language, and describe the written use case language. We define the building blocks

of the written use case language, starting from the use case set, via the use case and the various scenario types until we arrive at the action step. We analyse the action step down to syntactical building blocks.

2.2.1 Use Case

There are plentiful definitions for the term "use case". Jacobson's definition is elegant and succinct

> "[use case, n.] Specific way of using a system using some part of functionality" [57, page 154].

Some practitioners use the term *scenario* as an alias to the term *use case* when introducing the concept to business people.

Jacobson started designing systems with the help of use cases in the late 1960's [24, preface, page xx]. In 1992 Jacobson brought the technique to the attention of the developer community [57]. Jacobson's book, a presentation of object oriented analysis, highlights the dependency of developers on the ever evolving use cases [57, page 129]. Use cases might describe a business process, focus discussion about future software system, describe functional system requirements, or document a system's design [13, Page 58].

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 [67, page 298]. Use cases have been found to be also effective for generating test suites [31], and for generating security policies [38]. This success may arise because they tell coherent stories about how the system will be [24, Page 15]. Use cases follow a long human tradition of analysing scenarios [17].

Before we offer a formal definition of a use case and its constructs, we now give a brief overview of the subject. An activity is a description of a piece of work that forms one logical step within a process. An activity may be manual or automated [117, Page 13]. Cockburn advised authors of use case to apply restricted grammatical structure for the description of activities — subject, verb and object [24, page 90]. A scenario is a sequence of goal-achieving actions by various actors [24, Page 89], it is a mini specification consisting of partly ordered activities [24, Page 26]. An

extension is a stripped down use case that starts with a condition — the one that makes it relevant [24, page 99]. A scenario contains a sequence of action steps describing what happens under that condition. A scenario ends with delivery or abandonment of the extension goal [24, Pages 99-100]. A use case is a set of scenarios — a main success scenario and some alternative scenarios, each with its extensions [24, page 106]. For example, in Figure 2.7 on page 40 — UC Example we apply a use case language to describe a business process which we also articulate using BPMN in figure 2.2 on page 16 — BPMN Example. A use case set is a high level use case, of which some activities are further elaborated as lower level use cases, which in turn may be further elaborated, until sufficient detail is achieved.

2.2.2 Use Case Languages

The use case we described thus far is abstract. In this section we describe two of the many languages proposed to articulate use cases.

Jacobson's diagramming language for use cases [57, page 129] was included within the UML standard in 1997 [19, p.12]; it represents the connections between different use cases, and the actors that participate in the use cases, rather than expressing the content of a given use case [24, Page 233].

In many ways the UML use case diagram is a context diagram, showing the software system and the world around the system as illustrated in Figure 2.6 — Use case goal levels. In this big picture view, the system is treated as a closed box. In essence, the goal of the development process is to both fill and fulfill the box. The emphasis is on how the system relates to the world, not on its architecture or any other form of decomposition. Context diagrams have been around in one form or another for three decades and have been presented in various forms. Historically they were often shown as data flow diagrams (DFDs), which follows naturally from their use in structured analysis methods that make heavy use DFDs as a formalism [118]. In object oriented modelling the appropriate style is to express the system and the world around it as objects connected by associations. Michael Jackson has taken the concept of context diagrams further and used them as the basis for understanding problem frames [51].

UML context diagrams are a good tool to model the system boundaries,

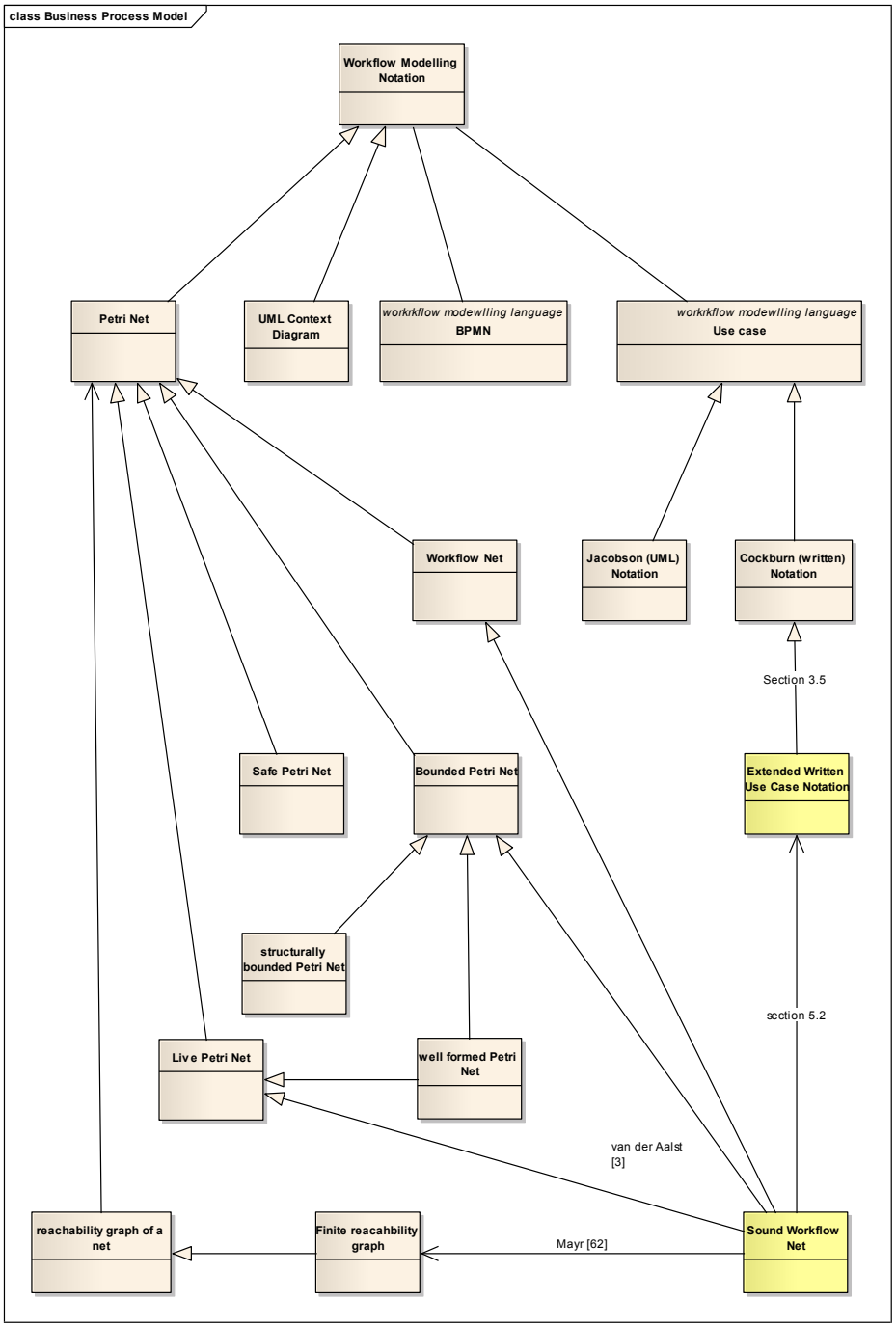


Figure 2.5 – A reference of the Business Process Management terminology introduced in this section

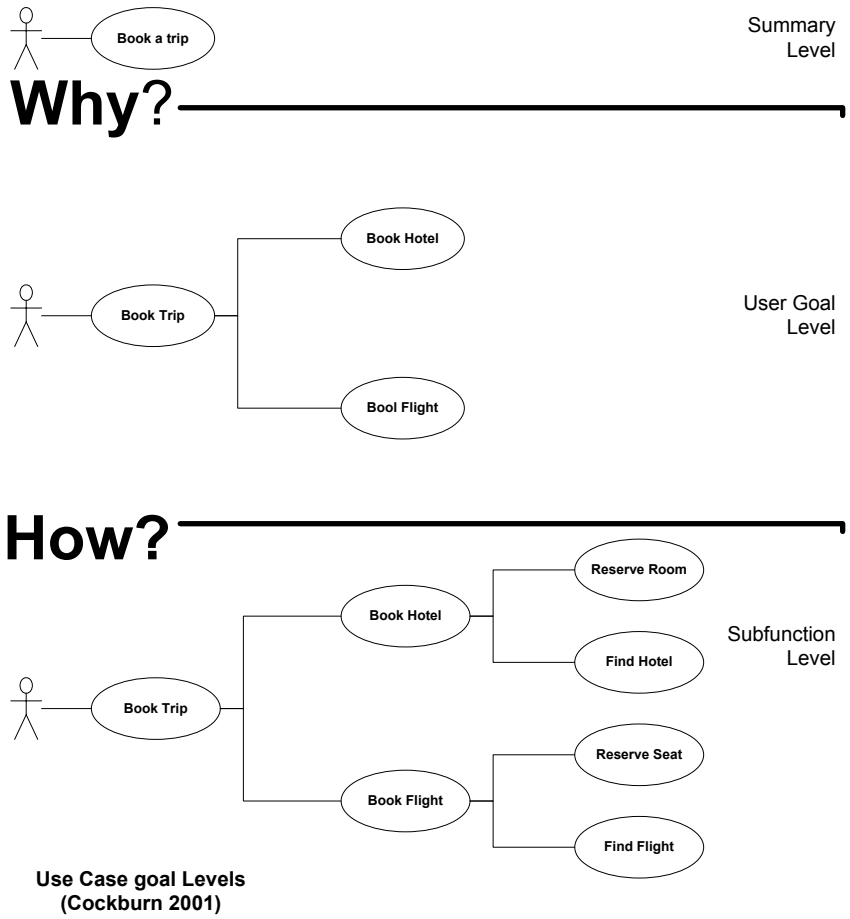


Figure 2.6 – **Use case goal levels** [24]. A requirements practitioner can break a use case into a use case set, exposing more information in the process.

identifying potential actors and use cases because it helps us focus on the things that interact with a system, while ignoring the services that those things require [13, p15].

Of the 28 use case languages Hurlbut surveyed [56], we adopted the written one, described by Cockburn [24] and elaborated by Cockburn *et al* [13].

Definition §2.23

written use case language, n. A use case language described by Cockburn [24] and elaborated by Cockburn *et al* [13].

In 2001, following anthropological observations of requirements practitioners over more than a decade, and conversations with Jacobson, Cockburn published the most visible work in the field: the monograph "Writing Effective Use-Cases" [24, Pages $xx - xxi$]. Guidelines for the usage of the textual language followed in 2003 [13]. Cockburn advocates readability and clarity, sometimes sacrificing precision and expressive power [24, pages 28 and 127].

Before we describe in detail the building blocks of the written use case language we provide two auxiliary definitions.

Definition §2.24

partial order, a. By a system is meant a set S together with a binary relation $R(x, y)$ which may hold for certain pairs of elements x and y of S. The relation $R(x, y)$ is read *"x precedes y"* and is written"$x > y$". A system is called **a partial order** if the following conditions are satisfied: if $(x < y) \Rightarrow (y \not< x)$; if $(x < y)$ and $(y < z) \Rightarrow (x < z)$ [35].

Cockburn, as we discuss later, identifies partial order as a simplification that makes the use cases easier to read [24, Page 26].

The written use case language has the following building blocks:

Definition §2.25

stakeholder, n. Someone or something that has a vested interest in the behaviour of the use case [24, page 53].

```
Use Case # 1 - ATM Cash Withdrawal
Level: Summary
Primary Actor: The customer who wants to withdraw cash.
Stakeholders: The ATM, The clearing System, The customer's bank.
Minimal Guarantee: The process will end with either a payment or a rejection.
Preconditions: The customer has an account and PIN.
Trigger: The customer pressed "Enter" on the ATM's keyboard.

Main success scenario:
1) The ATM request PIN.
2) The customer keys the PIN.
3) The ATM presents menu
4) The customer selects withdrawal.
5) The ATM asks for amount.
6) The Customer keys amount.
7) The ATM request approval from the clearing system.
8) The clearing system request approval from the customer's bank.
9) The customer's bank grants approval
10) The ATM dispenses the cash.

Extensions:
7)    The clearing system cannot find the customer's bank.
      7a. The clearing system informs this to the ATM.
      7b. The ATM terminates the session.

9)    The customer's bank does not grant approval.
      7a. The clearing system informs this to the ATM.
      7b. The ATM terminates the session.
```

Figure 2.7 – Basic written use case language example. The example is of equivalent content to the BPMN example in figure 2.2.

Stakeholders perform the activities in a use case while defending their own interests, in peruse of a specific and measurable goal which is of benefit to the primary actor.

Definition §2.26

primary actor, n. The stakeholder that calls on the system to deliver one of its services [24, page 54].

Definition §2.27

use case title, n. An active verb phrase that represents the goal of the primary actor of the use case [24, Back Cover].

Definition §2.28

action step, n. A labelled written use case line which has a restricted grammatical structure [24, Page 90].

The action step structure should be absolutely simple, as shown in Figure 2.8 on page 43 [24, page 90]. The role for the activity is the grammatical subject in the action step, and the activity is given by the predicate and the object.

Definition §2.29
action step ID, n. A label that uniquely identifies an action step within a written use case [24, Page 218].

Here we emphasise that action step IDs are labels, not sort fields as the action steps in a use case are partially ordered. It is easy to fall into the misconception that action step IDs are sorting fields, as they are monotonously growing.

Definition §2.30
specification, n. Specific, explicit, or detailed mention, enumeration, or statement of something [25].

Definition §2.31
scenario, n. [A mini specification built of] partly ordered set of action steps [24, page 90].

We introduced the terms action step and scenario in unison as practitioners often oscillate between an action step, and a scenario when describing one business process even while retaining the level of detail. When a business analyst discovers more details about a business process, the business analyst may split an action step into a scenario or add an extension, a scenario in its own right. *Vice versa,* when a business analyst simplifies a process, the business analyst may collapse a scenario into an action step.

Definition §2.32
use case, n. A collection of scenarios — a main success scenario, some alternative scenarios, and a multitude of extensions. In the general sense, though, extensions are miniature scenarios in their own rights [24, page 106].

Definition §2.33

use case set, n. Collection of activities in a high level use case that are further elaborated as lower level use cases, until sufficient detail is achieved. Cockburn's terminology is that of a use case and sub use cases [24, Chapter 10 and Page 219].

Again we introduce the terms use case and use case set in unison as business analyst may oscillate between a use case and use case set when describing one business process, even while retaining a level of detail. When a business analyst discovers more details about a process, the business analyst may split a use case into a use case set. *Vise verse,* when a business analyst simplifies a process, the business analyst often collapses a use case set into a use case.

If business analyst describes an action step elsewhere as a use case within the use case set, then the business analyst underlines the action step [24, page 113]. In Jacobson's UML language, on the other hand, there is no explicit distinction between use case, scenario and action step they are all annotated as ellipses. In Jacobson's UML language, the use case title is denoted within the ellipse. In a written use case language the name of the use case is the header of the written use case language . In the UML use case language goals are expressed on the arrows, in the written use case language, goals are part of the preamble to the use cases or the extension headers.

The following definition may do away with the term use case set, at the cost of clarity: *Use case is a set of use cases performed by various actors, each defending their own interests, in peruse of a specific and measurable goal which is of benefit to the primary actor.*

Definition §2.34

main success scenario, n. A top-to-bottom description of an easy-to-understand and fairly typical [sequence of events] in which the primary actor goal is delivered and all stakeholders' interests are satisfied [24, Page 87].

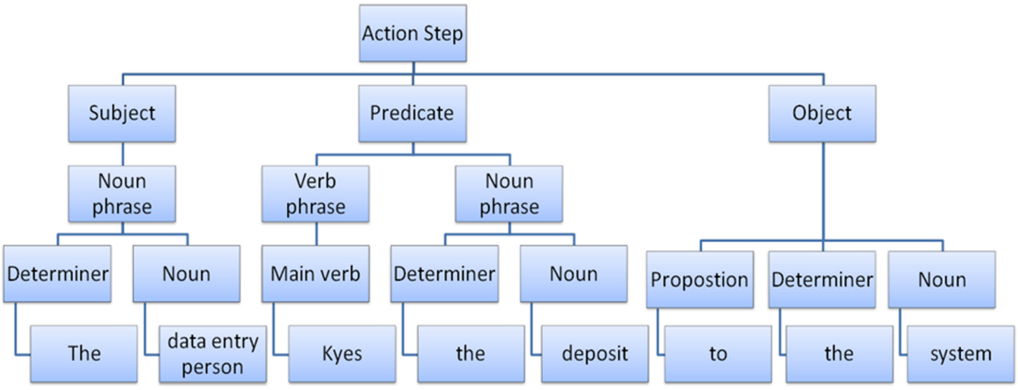


Figure 2.8 – The restricted grammatical structure of an action step. The role for the activity is the grammatical subject in the action step, and the activity is given by the verb and any grammatical direct or indirect objects, following [28, Page 252] and [24, page 90].

The main success scenario does not show any consideration for possible failures. If there are multiple success scenarios of similar complexity, the decision which scenario is the main success scenario is arbitrary.

Definition §2.35

extension, n. A stripped down scenario that starts with the condition that makes it relevant [24, Page 99].

Below the main success scenario reside story fragments that show what alternatives may occur. Each fragment thereafter has a header that describes a clearly detectable condition [13, p127]. These fragments later help developers to handle errors. These variations represent a significant, and possibly a majority portion of the effort involved in crafting use case sets.

2.3 Computer Human Interaction

In this section we look at the workflow modelling languages from the point of view of the Computer Human Interaction community. The Computer Human Interaction community has an annual conference — ACM CHI, the latest, CHI10 was conducted in April 2010 in Atlanta, USA. A key work in the HCI field is a book by Dix *et al* [34].

The human memory has a short term buffer from which an active decisions has to be undertaken if we are to migrate element to the long term memory. In a seminal work Miller estimated the capacity of the human short term memory to record seven plus or minus two elements [75]. The research within the CHI community is concerned with analysing the human physiology, how we receive information, how do we remember it, revisiting Miller's work on our memory capacity [75], how we solve problems, use our emotions, and the variation between individuals, the computer (memory and CPU) and the input output devices, as well the design process.

The work of Newell and Simon suggests that problem solving exercise has four steps: goal forming, operation selection, operation application and goal completion. The real power of problem space architecture is in recursion [34, Page 445].

Nielsen suggests that usability is about learnability, efficiency, memorability, errors, and satisfaction [81]. The definition of usability from ISO 9241-11 is becoming the main reference of usability, though. ISO 9241-11 defines usability as:

> **Definition §2.36**
> usability, n. The extent to which product can be used by specified users to achieve a specified goal with effectiveness, efficiency and satisfaction in a specific context of use [58].

> **Definition §2.37**
> usability metric, n. An observable and quantifiable way of measuring task success, user satisfaction or errors [15, page 7].

This observation might be simply noting that a task was completed successfully or noting the time required to complete a task. Evaluating a product with a very small sample size (without collecting any usability metrics) usually reveals most obvious usability problems [15, page 9] [34, Page 324]. Usability data can be cleaned up so that extreme values are not used in the analysis [15, page 12].

When comparing activities of subjects, two methods are common: within subjects and between subjects. In between subjects experiment design,

each participant is assigned a different condition. In within subjects experiment design, each participant is subjected to each different condition [34, Page 331].

Definition §2.38
within subjects, n. An experimental procedure when one compares different data for each participant [15, page 18].

A within subjects study does not require a large sample size and does not require to worry about differences between groups as each participant is compared to herself. However, one must worry about the carry over effect where performance in one condition impacts performance in another condition. A carry over effect might be the result of practice or fatigue. Counterbalancing the carry over effect may involve a change in the order in which different tasks are performed [15, page 18-19].

Definition §2.39
between subjects, n. An experimental procedure used to compare results for different participants [15, page 18].

Definition §2.40
mixed design, n. An experimental procedure that contains between subjects factors and within subject factors. [15, page 19].

Definition §2.41
independent variable, n. An experimental procedure used to compare results for different participants [15, page 18].

Definition §2.42
dependent variable, n. In an experiment the aspect that occurs as a result of the experiment [15, page 20].

Definition §2.43
ordinal data, n. Ordered groups or categories [15, page 21].

In usability, the most common occurrence of ordinal data comes from self reported data on questioners [15, page 21]. The statistical procedures that can be applied to ordinal data are frequencies, crosstabs, Chi square, Wilcoxon rank sum tests and Spearman rank correlation [15, page 23].

Definition §2.44
Interval Data, n. Continuous data where the difference between the measurements are meaningful but there is no natural zero point [15, page 22].

As a rule of thumb, one would treat data like interval data when a half point between any two of the defined data points makes sense [15, page 22 23].

Definition §2.45
Ratio Data, n. Interval data with the addition of an absolute zero [15, page 23].

There is not much difference between Interval Data and Ratio Data in terms of the available statistics. For Ratio Data one may use all descriptive data (including geometric means), $t - test$, ANOVAs, correlation and regression analysis [15, page 23].

2.4 Linguistics

The Linguistics community is mostly concerned with spoken or written languages, whether ancient or current, whether alive or dead, whether used by technologically–advanced societies or technologically–challenged societies, whether used by large group of people or very small groups, and whether used by societies that are in contact with speakers of other languages or by insulated societies. In this section we try to survey the highlights of the state of the art, noving from the physical level, to the syntactic level before arriving at the semantic level.

Rose at el, in a research programme whose description is outside the scope of this dissertation, have shown that language learning, which can be defined to a repeated exposure to a stimulus of information, is accompanied

by changed patterns of protein synthesis at relevant areas of the cortex [105, Page 302].

In the 1950s the social sciences were dominated by behaviorism, the school of thought popularized by John Watson and B. F. Skinner [92, Location 241]. Then came Chomsky, who worked earlier than Rose at el. Chomsky attacked what is still one of the foundations of twentieth-century intellectual life — the "Standard Social Science Model", according to which the human psyche is molded by the surrounding culture [92, Location 277].

Chomsky's first fundemental observation was that virtually every sentence that a person utters or understands is a brand-new combination of words, appearing for the first time in the history of the universe. Therefore Chomsky suggested that a language cannot be a repertoire of responses. The brain, Chomsky suggested, must contain a recipe or program that can build an unlimited set of sentences out of a finite list of rules.

Chomsky's second fundamental observation was that children develop these complex grammars rapidly and without formal instruction and grow up to give consistent interpretations to novel sentence constructions that they have never before encountered. Therefore, Chomsky argued, children must innately be equipped with a plan common to the grammars of all languages, a Universal Grammar, that tells them how to distill the syntactic patterns out of the speech of their parents [92, Location 244–9].

A significant question that caught the attenstion of the Linguistics field is whether thought is dependent on words. Do people literally think in English, Cherokee, Kivunjo, or, by 2050, Orwell's Newspeak? Or are thoughts couched in some silent medium of the braina language of thought, or "mentalese" — and merely clothed in words whenever we need to communicate them to a listener? No question could be more central to understanding the language instinct [92, Location 886]. The idea that thought is the same thing as language is an example of what can be called a conventional absurdity: a statement that goes against all common sense but that everyone believes because they dimly recall having heard it somewhere and because it is so pregnant with implications. We have all had the experience of uttering or writing a sentence, then stopping and realizing that it wasn't exactly what we meant to say. To have that feeling, there has to be a what we

meant to say that is different from what we said. Sometimes it is not easy to find any words that properly convey a thought. When we hear or read, we usually remember the gist, not the exact words, so there has to be such a thing as a gist that is not the same as a bunch of words. And if thoughts depended on words, how could a new word ever be coined? How could a child learn a word to begin with? How could translation from one language to another be possible [92, Location 910–14]?

The representations underlying thinking, on the one hand, and the sentences in a language, on the other, are in many ways at cross-purposes. Any particular thought in our head embraces a vast amount of information. But when it comes to communicating a thought to someone else, attention spans are short and mouths are slow. We end up with the following picture. People do not think in English or Chinese or Apache; they think in a language of thought [92, Location 1130-1136]. The way language works, is that each person's brain contains a lexicon of words and the concepts they stand for (a mental dictionary) and a set of rules that combine the words to convey relationships among concepts (a mental grammar) [92, Location 1398].

Chomsky suggests that the unordered super-rules (principles) are universal and innate, and that when children learn a particular language, they do not have to learn a long list of rules, because they were born knowing the super-rules. All they have to learn is whether their particular language has the parameter value head-first, as in English, or head-last, as in Japanese. They can do that merely by noticing whether a verb comes before or after its object in any sentence in their parents' speech. If the verb comes before the object, as in Eat your spinach!, the child concludes that the language is head-first; if it comes after, as in Your spinach eat!, the child concludes that the language is head-last [92, Location 1880–3].

When Chomsky introduced the terms in the behaviorist climate of the early 1960s, the reaction was sensational. Deep structure came to refer to everything that was hidden, profound, universal, or meaningful, and before long there was talk of the deep structure of visual perception, stories, myths, poems, paintings, musical compositions, and so on."Deep structure" is a prosaic technical gadget in grammatical theory. It is not the meaning of a sentence, nor is it what is universal across all human languages. Though

universal grammar and abstract phrase structures seem to be permanent features of grammatical theory, many linguists including, in his more recent writings, Chomsky himself thinks one can do without deep structure per se. The concept is actually quite simple [92][location 2061-2082].

Syntax is complex, but the complexity is there for a reason. For our thoughts are surely even more complex, and we are limited by a mouth that can pronounce a single word at a time. Grammar, a form of mental software, must have evolved under similar design specifications. Though psychologists under the influence of empiricism often suggest that grammar mirrors commands to the speech muscles, melodies in speech sounds, or mental scripts for the ways that people and things tend to interact, all these suggestions miss the mark [92, Location 2139]. Grammar is a protocol that has to interconnect the ear, the mouth, and the mind, three very different kinds of machine. It cannot be tailored to any of them but must have an abstract logic of its own [92, Location 2125-41].

Wittgenstein's work preceded Chomsky's. Wittgenstein spoke about surface and depth grammars. Chomsky spoke about 'deep structures', on the other hand, which are located 'far beyond the level of actual or even potential consciousness'. Wittgenstein worked is in the semantic layer which is above the syntactic layer in which Chomsky worked. Chomsky, in turn, worked in the syntactic layer which is above the physical layer in which Rose et al worked.

Wittgenstein, who opposed the deviation of systematic logic from ordinary language, investigated the borders between meaning and speech, between mathematics and languages, between a private and a public language, and between thoughts and words, in particular sensation words (notably 'pain').

Wittgenstein, accepting the existence of private language, placed communication in a marginal position interpreting between individuals. Noting that no two human beings share an identical associative context, Wittgenstein asked where, when, and by what rationally established criterion the process of free, yet potentially linked and significant, association in psychoanalysis could said to be said to have a stop.

As Wittgenstein argued in Philosophic Investigations, languages gen-

erate different social modes, and social modes further divide languages. Different linguistic communities literally inhabit different landscape of conscious being. We will want to come back to old truism. It touches the nature of language itself, on the absence of any satisfactory or generally accredited answer to the question 'what is language'. Indeed, text can conceal more than it conveys. Presiding Orwell, Wittgenstein foresees a 'speech therapy' that will mend the infirmities of ordinary language and the conflicts it provokes. [105, Pages 8, 62, 64, 91, 92, 97,105,169-77, 236, 290, 290n, 338, 497n].

Wittgenstein proposes that the meaning of a word is its use in the language. The Oxford English Dictionary, with its use base approach, can certainly say something about the use of the term language:

Definition §2.46

language,n [...] 2 a. The form of words in which something is communicated; manner or style of expression. 2 b. The vocabulary or phraseology of a particular sphere, discipline, profession, social group, etc.; jargon. 2 c. The style of a literary composition; (also) the wording of a document, statute, etc. [...] 6. The method of human communication, either spoken or written, consisting of the use of words in a structured and conventional way; (also) words [25, accessed October 2011].

A thread that is of concern to some members of the Linguistic community is the relationship between the expressive power of ones mother tongue and ones ability to perceive certain concepts. The now defunct Sapir-Whorf hypothesis states that deficiencies in ones mother tongue expressive power limit ones speakers' ability to express or understand concepts [33, page 150] and the alleged inability of native speakers of the Hopi language to grasp the notion of time is now infamous.

For example, the absence of the beloved German term "*Schadenfreude*"; for which some English speakers did not have an equivalent, until the publication of Supplement *III* to the OED in 1982; did not in any form prevent some of these native English speakers from indulging at times in the *malicious enjoyment of the misfortunes of others* [25, Accessed January 2011].

In contrast, the somewhat more accepted Boas Jakonson hypothesis, states that languages differ essentially in what must be conveyed and what may be conveyed [33, page 151].

For example, when one says in English that one spent a night with a friend, one has the option to revile the friend's gender — *"I stayed the night with my girl friend"* or not — *"I stayed the night with my friend"*. That is not the case in languages that require to state of the gender of the object of a sentence, such as Hebrew or German. Indeed languages such as Turkish, Finish, Estonian, Hungarian, Indonesian and Vietnamese have no grammatical gender at all [33, Page 200]. Hebrew and German forces the explicit specification of the gender of the friend but in English this data item is optional. This brings to the conclusion that the English writer does not need to think of the gender while speakers of other languages may have to.

2.5 Literacy

Literacy, with its high political and economical importance, is researched by communities including journalism, military personnel, education, brain research and psychology. While reading may look to some of us as a one activity process, reading is a complex set of activities that requires the gradual development of several skills. Here we describe the reading process and effort done some members to the community to assess the readability of various artifacts.

The language system's components are: Phonology, Semantics, Syntax and Discourse. A phoneme is the smallest unit of speech that distinguishes one word from the other. It is the fundamental element of the language system, the essential building block of all spoken words. Different combinations of just 42 phonemes produce the tens of thousands of words in the English language. The word cat, for example, consists of three phonemes: K, aaaa, and t. Before words can be identified, understood, stored in memory or retrieved from it, they must first be broken down into phonemes by the natural machinery of the brain. It is critical for both listening and reading [99, page 41].

The listener has to receive the phonemes at a sufficiently fast pace so that several can be held in the short term memory at the same time and integrated to form the intended words and phrases. Phonemes can be held in the short term memory only one or two seconds or about five unrelated words before each vanishes like a bubble. In producing a word, the human speech apparatus - the larynx, palate, tongue and lips - automatically compress the phonemes together. Consequently several phonemes are folded into a single pulse or bubble of sound, without any overt clue to the underlying segmental nature of speech. Hence spoken language appears to be seamless.

Both reading and writing is dependent on the assembling of phonemes, speaking is natural and reading is not. Functional MRI results visualise this clearly. For the object of the readers' attention (print) to gain entry into the language module, the reader must first convert the printed characters on the page into the phonetic code; otherwise these letters remain just a bunch of lines and circles totally devoid of linguistic meaning. As a person reads a sentence the person has to hold several bits of information in mind in order to put it all together and make sense of what he has just now read. He first decodes letters into sounds, then holds these sounds in his memory as he tries to decode the remaining letters in the word, and then he takes these stored sounds, blends them together, and forms a word. Words are stored [indexed] primary on the basis of their sounds, so the ability to hold words temporarily is a phonologic skill. Within the brain, the child is literally building the neural circuitry that links the sounds of spoken words, the phonemes, to the print code, the letters that represent these sounds

Phonological deficits are constant throughout the life of dyslexic man and woman. In dyslexic people the processing of phonemes is less well developed. A dyslexic child may say lotion when she means to say ocean. A dyslexic person may have a difficulty consciously. Paradoxically understanding dyslexia is important to the understanding reading as observing people with difficulties in reading highlights the mechanisms of reading, as fluent readers read in a hard to understand automatic manner. A seminal work in the subject is by Shaywitz [99]. The word dyslexia comes from the Greek words $\delta\upsilon\sigma$ - dys- ("impaired") and $\lambda\epsilon\xi\iota\varsigma$ lexis ("word"). Identi-

fied for the first time by Oswald Berkhan in 1881, the term 'dyslexia' was coined in 1887 by Rudolf Berlin, an ophthalmologist practising in Stuttgart, Germany. He used the term to refer to a case of a young boy who had a severe impairment in learning to read and write in spite of showing typical intellectual and physical abilities in all other respects. People with dyslexia are called dyslexic or dyslectic.

Dyslexia is a specific learning disability that is neurobiological in origin. It is characterised by difficulties with accurate and/or fluent word recognition and by poor spelling and decoding abilities. These difficulties typically result from a deficit in the phonological component of language that is often unexpected in relation to other cognitive abilities and the provision of effective classroom instruction. Secondary consequence may include problems in reading comprehension and reduced reading experience that can impede growth of vocabulary and background knowledge. Although dyslexia is the result of a neurological difference, it is not an intellectual disability. Dyslexia occurs at all levels of intelligence [69]. Dyslexia is a specific impairment in learning to read with a prevalence of 5% - 10% in school age children [90]. Spelling and reading are intimately linked; to spell correctly a child relies on his stored representation of a word, and these are imperfect in dyslexia. In fact, spelling errors may remain long after a dyslexic child or adult has learnt to decode most words accurately.

The reader's job is to convert the letters into their sounds and to appreciate that the words are composed of smaller segments of phonemes. Dyslexic children and adults have difficulty developing awareness that spoken and written words are comprised of phonemes. Overall, the reader must come to know that the letters on the page represent, or map onto, the sounds one hears when the words are spoken. These linkages are referred to as the alphabetic principle. Beginning readers must first analyse a word; skilled readers indentify words instantaneously, by sight. fMRI imaging studies revealed markedly different brain activation patterns in dyslexic readers compared to those in good readers. As they read, good readers activate the back of the brain, where the language reside, and also to some extent the front of the brain. In contrast, dyslexic readers show a fault in the system: under activation of neural pathways in the back of the brain.

At all ages good readers show a consistent pattern: strong activation in the back of the brain, with lesser activation in the front. In contrast brain activation in dyslexic readers show increased activation in the frontal region in order to compensate for the disruption in the back of the brain. One means of compensating for a reading difficulty, for example, is to sub vocalise (say words under your breath) as you read, a process that utilised a region in the front of the brain responsible for articulating spoken words. Reading problems do not go away, they are persistent and now we know why.

One of the problems in public education and mass communication is how to tell whether a particular piece of writing is likely to be readable to a particular group of readers. Two major solutions are possible: measuring and predicting readability. Measuring, by judgements or tests, involves using readers. Predicting by readability formulas, does not involve readers but instead uses counts of language elements in the piece of writing. Judgements and comprehension tests, on the other hand, are not predictive devices in this sense. It is a predictive device in the sense that no actual participation by readers is needed [61].

Definition §2.47

readability formula, A mathematical equation derived by regression analysis. This procedure finds the equation which best expresses the relationship between two variables, which in this case are a measure of the difficulty experienced by people reading a given text, and a measure of the linguistic characteristics of that text. This formula can then be used to predict reading difficulty from the linguistic characteristics of other texts [71].

There are numerous documented readability formulas, indeed Klare surveyed more that 50 [61]. Popular readability formulas are FOG [61, Page 89], SMOG [61, Page 79] and Flesch Kincaid. The FOG readability metric is defined as:

$$GL_{FOG} = 3.0680 + 0.877 * ASL + 0.984 * PoM$$

where GL_{FOG} = FOG Grade Level and ASL = Average Sentence Length and PoM = Percentage Of Monosyllables [101]. The SMOG readability

metric [3] defined as:

$$GL_{SMOG} = 3 + \sqrt{NP}$$

where GL_{SMOG} = SMOG Grade Level and NP = Number of Polysyllable Words in 30 Sentences. If the document is longer than 30 sentences, the first 10 sentences, the middle 10 sentences, and the last 10 sentences are used. If the document has fewer than 30 sentences, some rules and a conversion table are used to calculate the grade level. The Flesch-Kincaid readability metric is defined as:

$$GL_{FK} = 0.39 * AW + 11.80 * AW - AS * 15.5$$

where GL_{FK} = Flesch-Kincaid Grade Level, AW = Average Number of Words Per Sentence and AS = Average Number of Syllables Per Word [101].

SMOG, whose origin is in the pre-word processing journalism, can be calculated manually. In [71], McLaughlin selected 8 articles and called for three specialists in literacy training to make content analyses and identify the ten most important ideas in each passage. After reading a passage each reader was asked to recall its entire content as fully as possible. This method of unaided recall was used in order to avoid the prompts which are inevitably given to a subject when he is questioned directly. The recalls were tape recorded and later transcribed verbatim. Each transcript was then compared with the list of ten main ideas, so that every passage was rated for comprehension by each reader on a scale of 0 to 10, McLaughlin found that there is a perfect negative rank correlation between polysyllable counts and the measures of reading efficiency.

The biggest body of work we found on evaluation of readability is in the medical field. This may be because readability can be a matter of life and death, maybe because informed consent is enforced by law, or maybe because the medical communities is accustomed to statistical based research. For example Davis *et al* [30] compared the readability of two consent forms using a verbal interviews of 183 adults recruited from private and university oncology clinics and a low income housing complex adults recruited from private and university oncology clinics and a low income housing complex. Davis *et al* asked ten question such as "What is the purpose of the consent form?" or "What is chemotherapy?". Their conclusion is that an illustrated

consent form was more readable than a consent form that did not include illustrations [30].

2.6 Reflections

The research I have described above leaves several gaps in important issues that need to be addressed when we consider the proposal to configure a workflow engine by use cases. Below I focus on gaps in the literature that lead to the hypotheses whose answers make up the reminder of this dissertation.

Written use case languages can be *ambiguous* as they use natural language. There seems to be an evident trade-off between the ease of a communication with business experts and accuracy. To overcome the natural language ambiguities, Cox *et al* show how to improve the quality of use cases with checklist driven reviews [27]. Törner *et al* suggest that it is possible to increase the correctness, consistency, completeness, readability and level of detail as well as to avoid ambiguity [108]. Nonetheless, Cockburn applied partial ordering to the action steps in the written use case language, consciously sacrificing expressive power for the sake of readability [24, Page 26].

> *Is the sacrifice needed? What should be done to resolve this limitation? Will overcoming the partial ordering limitation reduce readability? Will a fully ordered extended written use case language be readable?*

The current approach to workflow configuration is to replace the use cases that were formed in the requirement phase with more accurate languages such as BPMN or the UML activity diagram as projects move from the analysis phase into the design phase.

> *Is it possible to gradually extend artifacts using an extended written use case language as they flow from the analysis stage into design stage rather than rewrite then all together?*

While the written use case language is a popular requirement elicitation language, it has not been proposed to act as workflow modelling language.

Indeed, until mid 2010 I have not seen any other explicit or implicit reference to the idea of use case oriented workflow engine, an idea my supervisor and myself published in 2007 [86]. I have found the first implicit reference to our idea in a 2010 book by Fiammate [40]. In particular [40, Chapter 5] describes the dynamic assembly of processes from predefined use cases tree.

> *Will an extended written use case language, that is fully ordered be expressive enough to be a useful workflow modelling language?*

Lee *et al* bridged the Business Process Management and the Requirement Engineering communities by contributing a mechanism to translate use cases into Petri nets, enabling the mathematical verification of use cases [66]. We did not find any literature covering the converse.

> *Is it possible to demonstrate that every sound workflow net can be translated into use case set?*

It is commonly held that diagrams and visual languages are easy for humans to understand; a seminal argument for the benefits of visual language over textual language was given by Larkin and Simon [64], who reasoned that text is limited to a linear order, whereas a diagram allows more information to be carried by the spatial arrangement of different elements in the language. In contrast to Larkin and Simon's arguments, Moher *et al* [76] looked at several ways to express program structures in text and in diagrams (Petri Nets); they found "for our tasks, graphics were no better than text, and in several cases were considerably worse".

> *Who is right? Larkin and Simon? Moher et al? Or; should we present both a graphical language and a written language to achieve the highest readability? And if so, does the order count?*

I expected that readability metrics would have been sufficient to assess readability, however, I found that most of the readability metrics ignore document content and only consider surface linguistic features. Some surface linguistic features, such as average number of words per sentence, are influenced by presentation style, and some monosyllable words, such as "quark",

represent concepts that are not easy to understand. Features based on sentence length and number of syllables per word are common in readability measures such as the FOG, SMOG and Flesch Kincaid metrics, presumably based on the hypotheses that longer sentences, and sentences containing longer words, are more difficult to read [100].

Having read [71], one may claim that McLaughlin had only demonstrated that people can better answer questions on the subject of *"saying no to your children"* than about the subject of *"the meaning of organ transfer"*, rather than demonstrate causal relationship between the SMOG rating of an artifact and its readability. As McLaughlin did not assess the initial domain knowledge of participants, his research, one may claim, does not solely attributes the measured knowledge to the readding of instrument, but to initial domain knowledge. Furthermore, Davis *et al* [30] did not guarantee information equivalence between artifacts.

> *Are these methodological faults? If so, is it possible to ethically overcome these methodological faults when experimenting on consenting human beings?*

Chapter 3

Architecture

> A stone the builders had despised and it has risen and it became a cornerstone.
>
> Psalm 118:22

Here we articulate the main tenet that is explained in this dissertation — the proposal to feed use cases directly into a workflow engine as a workflow modelling language. In this chapter our focus is on the overall design of a workflow engine that accepts this language. Within the context established in Chapter 2, we first identify two classes of ambiguities inherent to the written use case language; one class relates to parallelism, another to choice. We explain why Cockburn knowingly accepted these ambiguities, and explain why we had to resolve them. We then describe the external interfaces of a use case oriented workflow engine, concentrating on properties and methods. We define the term use case oriented workflow engine, the cornerstone of this dissertation, and in conclusion, describe how various stakeholders interact with a use case oriented workflow engine.

Having articulated our proposal in this chapter we use the remaining chapters of this dissertation to evaluate this proposition by studying in detail a real life implementation of a use case oriented workflow engine, by surveying the variety of business processes that can be enacted by a use

case oriented workflow engine, and by empirically assessing the readability of the written use case language.

3.1 Methodology

The methodology we employ in this chapter is that of logical design, without covering in detail most implementation issue. We describe the main aspects of a workflow engine taking use cases as models, without entering into implementation details, give details of the input language, and build the object model of a use case oriented workflow engine, articulating classes, methods and properties. We conclude by describing the roles of various stakeholders in an organisation that deploys a use case oriented workflow engine.

The logical design helps us concentrate on the algorithmic core of the use case oriented workflow engine. The description we present is abstract, succinct, and platform agnostic. It is written in a way that may survive the test of time. More details of an actual implementation are in section 4.5 — Implementation on Off The Shelf Engine

A limitation of this approach is that it does not describe in detail implementation issues such as: document management, user interface management, messaging, caching, authentication or database management. A would-be implementer will need to supplement this architecture with a physical design. In particular, this would-be implementer will have to choose one of the many platforms one could leverage for the provisioning of a use case oriented workflow engine, whether a traditional off-the-shelf workflow engine from the likes of FileNet, TIBCO or K2, or other platforms, such as Microsoft's SharePoint or the fundamental .Net or J2EE platforms which may be leveraged just as successfully.

3.2 The Enactment Round Trip

In this section we discuss the overall application of a workflow engine as it applies to one configured by use cases.

As illustrated in Figure 3.1 — State Transition, the operation of a workflow engine may be split into three interweaving states — modelling and ver-

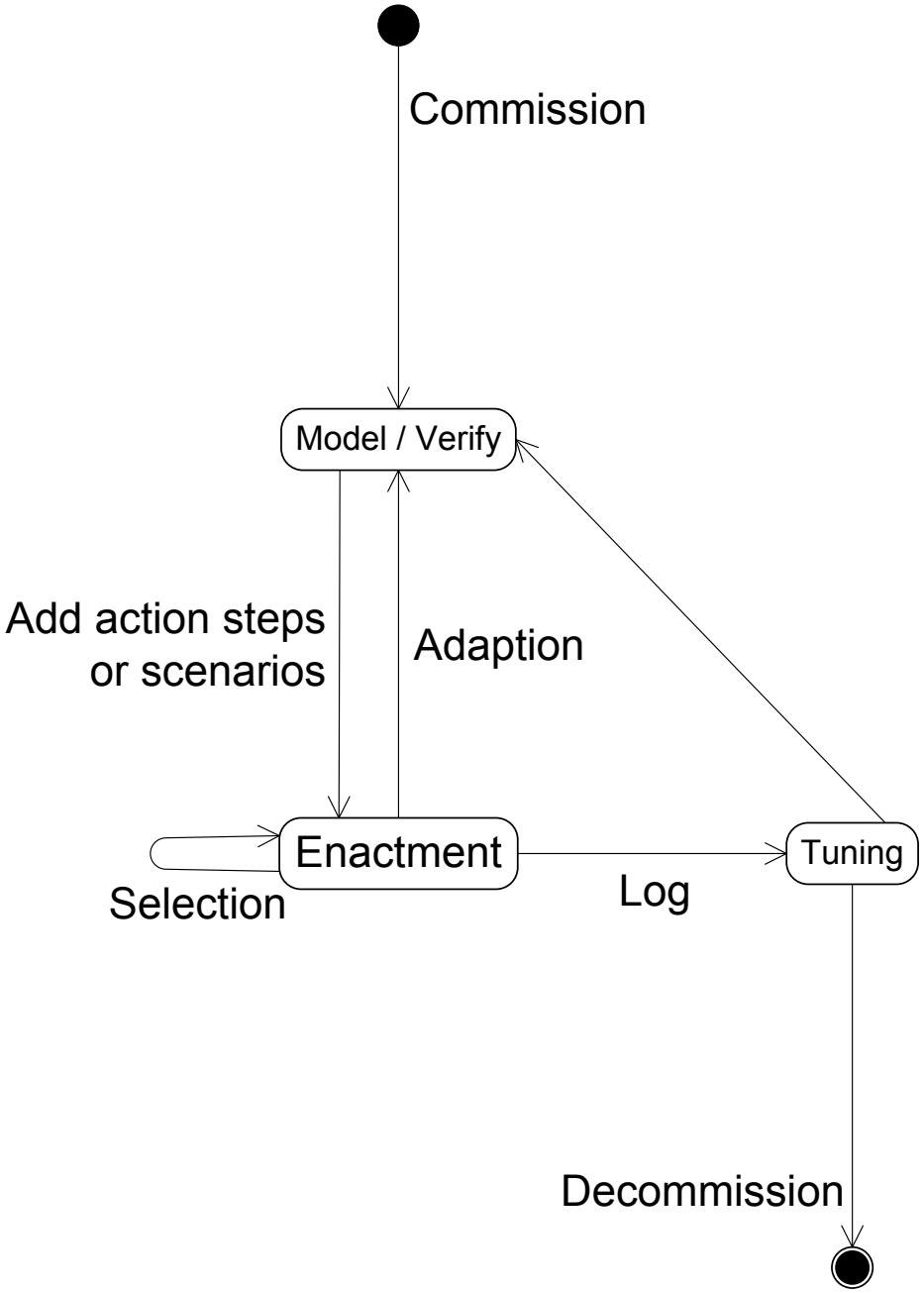


Figure 3.1 – **State Transition** — The enactment round-trip

ification, enactment, and tuning. At modelling time workflow participants, business analysts and workflow configuration officers model the process, verify the model and feed the model into the workflow engine. At run time the workflow engine guides the workflow participants through the enactment of business processes and monitors the processes. After the event, management discovers opportunities for process improvement by mining knowledge from the business activity monitor, and tunes the business processes [9]. The acceleration of this cycle may be desirable in dynamic environments.

While all workflow engines should follow these transitions between the states, with a use case oriented workflow engine the transition between the three state is *dynamic*. The platform allows the two flexibilities that Heinl *et al* required from a workflow engine, namely: flexibility by selection and flexibility by adaptation [53].

A major advantage of the use case oriented workflow engine over the incumbent engines is that the movement within and between the states is very quick. At design time, business analysts articulate a business process as a use case set. The business analysts then the use case set to the workflow configuration officer. The workflow configuration officer extends the use case set, in a manner that we describe below. The workflow configuration officer then creates a business transaction routing sheet from each scenario and from each extension to each use case, each business transaction routing sheet group, subject to an attribute observed, the attribute being the trigger for the use case or the trigger for the extension.

At run time, when a work item arrives at the organisation, the first workflow participant to touch the work item catalogues the work item, and then lists the attributes of the work item. Using these attributes the workflow engine links business transaction routing sheets to the work item.

When a workflow participant requests the next worklist, the use case oriented workflow engine, with a collection of work items on hand, makes a dispatching decisions based on (*i*) attributes of the work items on hand, (*ii*) the roles of the workflow participant, (*iii*) the prioritisation policies of the organisation and (*iv*) other business rules. Following this, the workflow participant performs action step by action step according to use case scenario captured in the business transaction routing sheets, until eventu-

ally an activity is found that the workflow participant cannot deal with, at which point the workflow engine passes the work item to another workflow participant. As the workflow participant executes each activity, the workflow participant acknowledges this to the workflow engine. From time to time the workflow engine records audit data to the business activity monitor describing the work item's attributes and progress through the action steps. Eventually the processing of the workflow item terminates and the workflow item is closed.

After the event, management monitors the attributes logged by the business activity monitor in order to understand process performance. From time to time management uses these observation to fine tune the process. The use case oriented workflow engine logs three types of massages into the business activity monitor: (*i*) work item massages, (*ii*) activity messages, and (*iii*) work item properties massages. The use case oriented workflow engine logs these messages at five occasions: (*i*) when work items are created (*ii*) when work items are terminated, (*iii*) when activities are spawned, (*iv*) when activities are completed and (*v*) when observations are assigned to work items. The work item massages keys are the indexing elements of the work items. The activity messages contain the names of the action steps that were undertaken on the work item. The work item properties messages take the observation that the pilot applied to the work item.

The difference in the time span between the creation of a work item and the termination of a work item represent the service visible to customers (External SLA). The difference in the time span between the spawning of an activity and the termination of an activity represent the service level offered by a department (Internal SLA). The presence of some observations can be used for tuning purposes as well.

As illustrated in Figure 3.2 on page 65, the written use case language lends itself naturally to this mode of operation because roles (such as primary actor or stakeholder) or scenarios (such as main success scenario, extension or alternative scenario), the core building blocks of the grammar of the language. The actor represents one of the roles a workflow participant may have. The extension header describes the attributes of the work item and the scenario list the workflow activities. When an end user, or a com-

puterised system, indicates to the use case oriented workflow engine which observation are met, the workflow engine is in a position to dispatch the appropriate work items to the appropriate role bearer, listing the required activities.

3.3 Extending the Input Language

Use cases are widely employed as a requirement elicitation tool, being accepted by both IT professionals and representatives of end users. As use cases are commonly produced during the requirements elicitation stages of the software delivery life cycle, they are available to the workflow configuration officer *naturally* [67, Page 297]. Cockburn designed his written use case language with requirement elicitation in mind, to be used in a phase in a software delivery life cycle where some uncertainty is tolerated, even encouraged, where solutions are not articulated, and optimisation is not yet needed. As such, they lack the expressive power to describe the fine nuances of parallelism and choice; using Moody's terminology, the written use case language suffers from symbol deficit [77]. We, on the other hand, apply Cockburn's language for workflow modelling, an application that requires a higher level of precision, and thus more expressive power. For its role as a workflow modelling language, we extend the written use case language in small ways, described here.

The written use case language's action steps are partially ordered. Cockburn argues that partial ordering is sufficient for requirement elicitation [24, Page-26] and we agree with him. However, when taking the written use case language into a new domain, to which it was not originally designed, and utilising the language as a workflow modelling language, the written use case language has to be extended so it can have the expressive power needed to articulate the order of action steps. The challenge is to maintain the business oriented nature of the language, while eliminating the ambiguities. That is a trade off between usability and expressive power [34, Page 606].

In this section we highlight ambiguities inherent to the written use case language and then we propose an extension to the language's syntax and

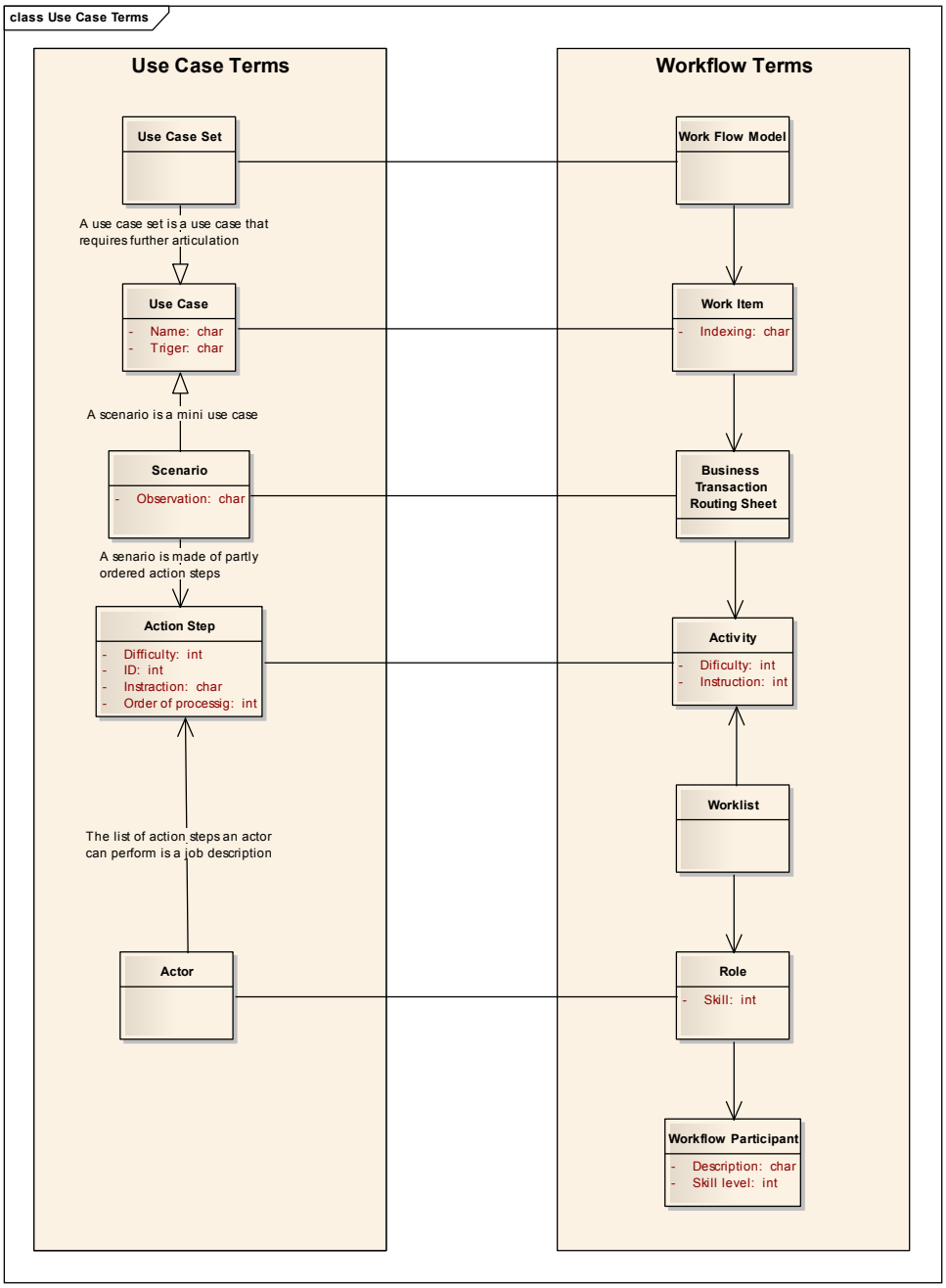


Figure 3.2 – Reference model of the use case terminology. Please note that the order of processing identifier and its parallelism implications are the only extension we introduce to Cockburn's language.

a style guide element, that together would resolve the ambiguities and so make the language suitable for configuring a workflow engine. We show that to resolve the ambiguities all the workflow configuration officer has to do is to extend the use case set as we describe below. In section 5.3 — Sound Workflow Nets Approach we provide a proof that with these extensions, the written use case language is sufficiently expressive.

Recall that the action step IDs in the written use case language are labels that uniquely identify conditions and activities, and that they are not order of processing identifiers. As we illustrate in Figure 3.3 on page 68 and in Figure 3.4 on page 69, the price of the partial ordering of action steps in a scenario is a high level of ambiguity. In these figures, inspired by [96, 115], we present workflow patterns articulated in BPMN in the left column and the same workflow pattern articulated as our extended written use case language in the right column.

Due to the partial ordering of action steps, use case #1, located on top of Figure 3.3, is ambiguous. The workflow configuration officer may interpret use case #1 as any of the six BPMN diagrams listed below it as the nature of the parallelism of activities A, B and C is yet to be articulated. The written use case language shown are identical except for the added order of processing identifier which serve to distinguish the allowed ordering of action step.

The workflow configuration office may wonder whether activities A, B and C are sequential, as articulated in use case #2 (WCP01-Sequence). Or whether activity A comes first, and then when A is completed, both activities B and C are executed concurrently, as articulated in use case #3 (WCP02-Parallel Split). Or, *vice verse*, do activities A and B occur concurrently, and only when both are completed does activity C commence, as articulated in use case #4 (WCP03-Synchronisation)? Or maybe activity C can start only after either activity A or activity B complete (but not necessarily both of them), as articulated in use case #5 (WCP05-Simple Merge). Perhaps all three activities should be executed concurrently, as articulated in use case #6? Or should activity C be executed while activities A and B are executed one after the other, as articulated in use case #7? Or, does the order not count at all?

In the same way, use case #8, located at the top of Figure 3.4 can be interpreted as one of the two BPMN diagrams listed below it. The reason for that ambiguity is that in use case #8 we do not articulate explicitly if conditions c' and b' are mutually exclusive or if conditions c' and b' are potentially satisfied together.

We resolve the ambiguity illustrated in Figure 3.4 by recording the verification activity explicitly as an action step and by articulating all the expected results as extensions that are nested (exclusive) or at the same level (potentially coexisting). It is natural to describe an inspection as an activity, for example, in the medical industry a blood test is an activity with price, duration, and many possible results (some of which may be unexpected). We anticipate that one of the roles of the workflow configuration officer would be to help end users with that articulation.

Definition §3.1
to gild the lily, v. To embellish excessively, to add ornament where none is needed [25], such as adding this definition.

Being acutely conscious that we are about to gild the lily, and reduce the usability of the written use case language, we now extend the written use case language by placing some extra information about the "order" of action steps.

Definition §3.2
order of processing identifier, n. An integer that defines the sequence of action step within a written use case.

Generally, the order of processing identifier is a monotonically increasing positive integer. However, when the order of some action steps is of no importance, or when action steps are taken in parallel, then these action steps share an order of processing identifier. Synchronisation steps (merges) are denoted as action steps whose order of processing identifiers are bigger than that of the synchronised action steps. When several streams of activity start (junction), each stream is represented by an individual sub use case, each represented by a single underlined action steps [24, Page 113], or by a adding a decimal point and a number to the order of processing identifier.

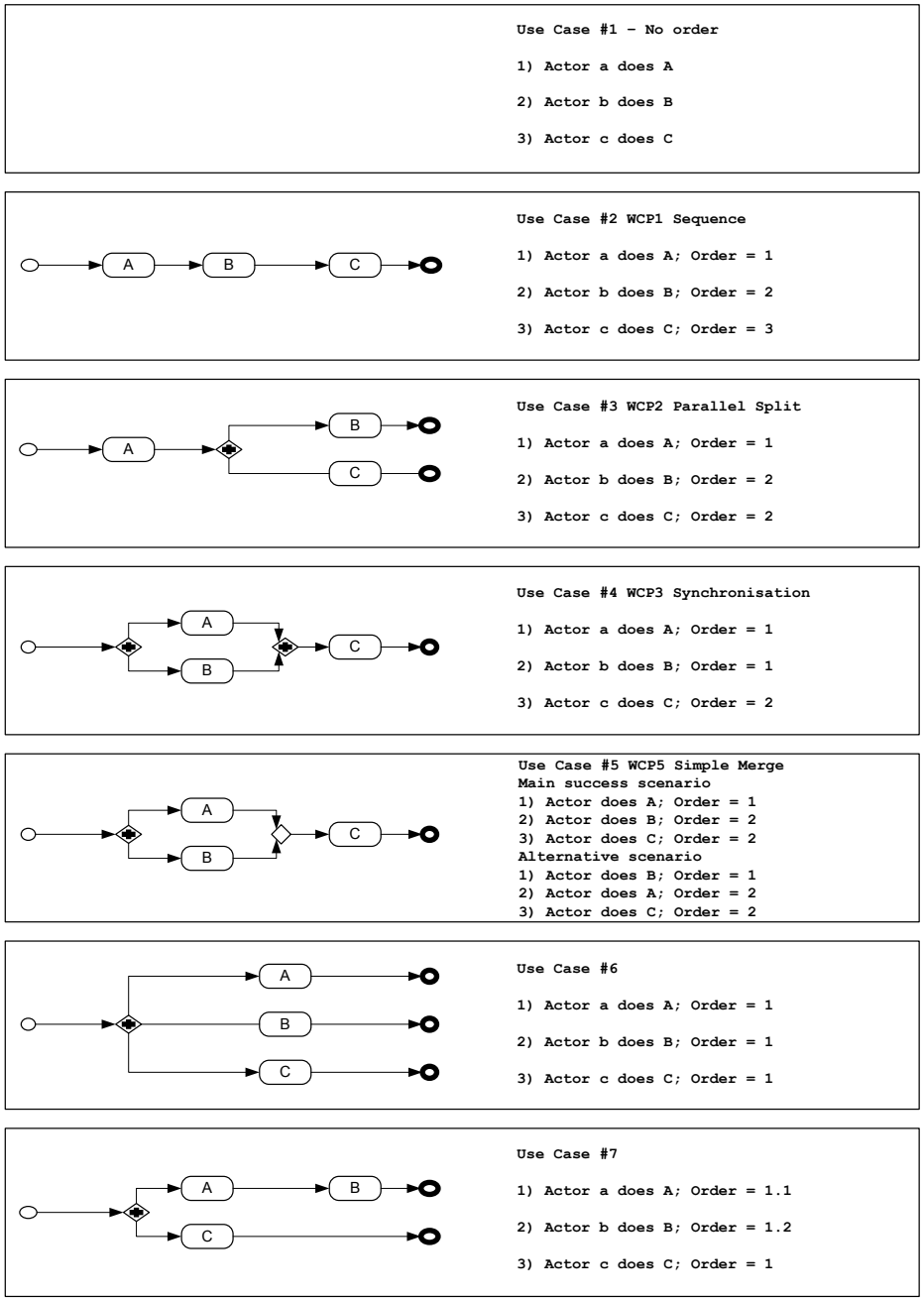
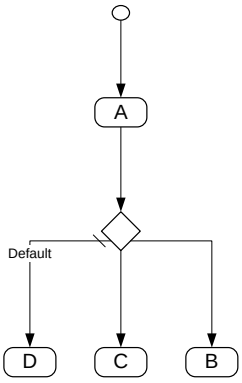
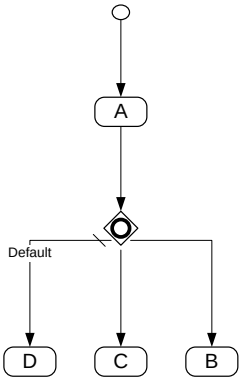


Figure 3.3 – **Partial Order.** Due to the partial order of action steps, use case #1 can be interpreted in six different topologies. When supplemented with order of processing identifier, each use case can only be interpreted as per the adjacent BPMN diagram.

```
Use Case #8 WCP4 or WCP6

Main success scenario
1) Actor does A;        Order = 1
2) Actor does D;        Order = 2
3) The use case ends;   order = 3

Extensions
2a) Condition c'
    2a1) Actor does C;  Order = 2

2b) Condition b'
    2b1) Actor does B;  Order = 2
```




```
Use Case #9 WCP4 Exclusive choice

Main success scenario
1) Actor does A;        Order = 1
2) Actor performs test; Order = 2
3) The use case ends;   order = 4

Extensions
2a) Condition d'
    2a1) Actor does D; Order = 3

    2a1a) Condition b'
          2a1a1) Actor does B; Order = 3

          2a1a1a) Condition c'
                  2a1a1a1) Actor does C; Order = 3
```




```
Use Case #10 WCP6 Multiple
choices

Main success scenario
1) Actor does A;        Order = 1
2) Actor performs test; Order = 2
3) The use case ends;   order = 4

Extensions
2a) Condition d'
    2b1) Actor does D;  Order = 3

2b) Condition b'
    2b1) Actor does B;  Order = 3

2c) Condition c'
    2c2) Actor does C;  Order = 3
```

Figure 3.4 – Ambiguous choice patterns

We support three notions of parallelism. (*i*) When action steps must be taken in parallel, their identical order of processing identifiers are supplemented with the ‖ sign. (*ii*) When action steps may be taken in parallel or in any sequence, their identical order of processing identifiers are supplemented with the \\ sign. (*iii*) However, as our work originated in office environment, where the workflow engine locks supporting documents to workflow participants for exclusive use, partial order is implicit. That is, the Workflow Pattern Coalition's pattern WCP-40 is the implicit default. The pattern is described as

> Each member of a set of activities must be executed once. They can be executed in any order but no two activities can be executed at the same time, i.e. no two activities can be active for the same process instance at the same time. Once all of the activities have completed, the next activity in the process can be initiated [96, Page 73].

As the reader may find that the definition of the order of processing identifier requires further elaborations we provide examples with the help of patterns drawn from the Workflow Patterns Coalition's library [96]. We will return to this list when we explore the expressive power of the extended written use case language in section 5.2 — Patterns Approach

(a) In general, the order of processing identifier is a monotonically increasing integer.
E.g.

- in workflow WCP-01 in section B on page 207 the sequential, action steps 'A' and 'B' have '1' and '2' as their order of processing identifiers respectively.

(b) IF the order of some action steps is of no importance,
OR the action steps are parallel,
THEN these action steps share an order processing identifier.
E.g.

- In workflow WCP-02 in figure 5.3 on page 122, the parallel action steps ‘A’ and ‘B’ share ‘2’ as their order of processing identifiers.
- In workflow WCP-13 in figure B on page 226, the six parallel action steps ‘1’ to ‘6’ share ‘1’ as their order of processing identifiers.

(c) Synchronisation steps are described by an action step whose order of processing identifier is bigger than that of the proceeding parallel action steps.
E.g.

- in WCP-05 in figure 5.3 on page 122, in the main success scenario, action step ‘C’ with ‘3’ as order of processing identifier, joins steps ’A’ with order of processing identifier ‘2’ and ’B’ with order of processing identifier ‘2’.

- In workflow pattern WCP-40 in figure 5.2 on page 119, action steps ‘B’,‘C’,‘D’ and ‘E’ are parallel (or they can be performed in any order) as the action steps share ‘2’ as their order of processing identifier.

- In workflow pattern WCP-07a, described in figure 5.4 on page 123 the sequential transitions Transition-B and Transition-D, have $order = 2.1$ and $order = 2.2$ respectively, while Transition-C, the single transition in the parallel stream, has $order = 2$.

- In workflow pattern WCP-03 in figure 5.3 on page 122; Transition-C, with $order = 2$, synchronises after Transition-A and Transition-B, both with $order = 1$.

.

We are now equipped to define our workflow modelling language as follows:

Definition §3.3
extended written use case language, n. A workflow modelling

language that supplements the written use case language by: (*i*) adding order of processing identifier to each action step, (*ii*) indenting mutually exclusive extensions, and (*iii*) listing potentially satisfied together extensions right below each other.

This way of presenting business processes has some features of natural languages and some features of programming languages. The term language is not a perfect match. Nonetheless, we see it is reasonable to refer as languages to the presentation formats of business processes, whether with the help of use case description syntax, or with the help of BPMN. Indeed other presentation formats such as UML and SQL are called languages.

3.4 Use Case Oriented Workflow Engine

Here we describe a use case oriented workflow engine as a black box that exposes properties and methods. In section 4.5 on page 104 — Implementation on Off The Shelf Engine we lift the veil and describe a few details of the implementation of a specific use case oriented workflow engine leveraging IBM's FileNet Panagon APIs.

3.4.1 Properties

Here we provide, using figure 3.5 on page 75, a reference model of a use case oriented workflow engine. In section 3.4.5 on page 80 — The Role of the Workflow Configuration Officer we describe an algorithm we developed to infer the data that the workflow configuration officer needs to populate from use case which business analysts produce during requirements elicitation stage. In section 4.5 on page 104 — Implementation on Off The Shelf Engine, we describe how to build a use case oriented workflow engine above a general purpose workflow engine.

The first interface we describe allows the use case oriented workflow engine to instruct the workflow participant to perform an activity.

Definition §3.4
instruct, n. To furnish with authoritative directions as to action; to direct, command [25].

Definition §3.5

instruction, n. An interface used to instruct a workflow participant to perform an activity.

Each activity has an attribute named instruction. Later we explain how the workflow configuration officer infers the instructions from use cases.

Definition §3.6

business transaction routing sheet, n. An ordered set of activities that workflow participants perform as a group.

A business transaction routing sheet has one or more activities. An activity, though, belongs to one and only one business transaction routing sheet. A business process can have one or many business transaction routing sheets. Just like the routing used in production floors to describe the processes an order has to pass, so does the business transaction routing sheet describe the activities that have to be executed as a group to fulfil part of the process. A work item has none, one or many business transaction routing sheets. For example of business transaction routing sheets derived from an extended written use case language please refer to Figures 3.6 — Sample Process and 3.7 — Business Transaction Routing Sheet.

The second interface we describe allows the workflow participant to describe properties of the work item so that the use case oriented workflow engine will be able to instruct other workflow participants to perform the needed activity.

The act of prescribing, or linking, business transaction routing sheets to a work item is called piloting (more about piloting in section 3.4.8 — The Role of the Workflow Participant when we explain the various roles involved in the enactment of a use case oriented workflow engine) . A work item that has no business transaction routing sheet assigned is a piece of information that arrives to the organisation and requires no action.

Definition §3.7

observe, v. To take note of or detect scientifically; to watch or examine methodically, esp. without experimental or therapeutic intervention; to perceive or learn by scientific inspection or measurement; [...] [25].

Definition §3.8
observation, n. An attribute of a work item one may observe.

Each business transaction routing sheet has an attribute named observation. The workflow configuration officer infers the observations from use case 's trigger and from the extension's condition.

A modern workflow engine is expected to present to the workflow participants the information they need to perform each activity.

Definition §3.9
supporting document, n. External artefact that corroborates a business transaction.

A work item may be associated with supporting documents that guide the workflow participants through the enactment of the workflow, or that are used as an evidence that the workflow was duly enacted (for example, a parking offence cannot be processed without a parking ticket signed in a certain form by a parking inspector).

In the 1980s, when imaging systems were promoted as workflow engines, one supporting document was provided with one work item; a $1:1$ ratio. Nowadays, any ratio is acceptable. $0:1$ ratio implies that the work item is not initiated by a supporting document, *e.g.* the periodic review of an investment portfolio is triggered by the arrival of a predefined date, rather than the arrival of a document. $n:1$ ratio implies that the several supporting documents are required for a work item to complete, *e.g* two signed documents are required, one by the selling party and one by the buying party before processing may commence. $1:n$ ratio implies that the 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 scanned paper to XML. Workflow participants no longer have to be humans. Workflow participants can now be automated computer programmes.

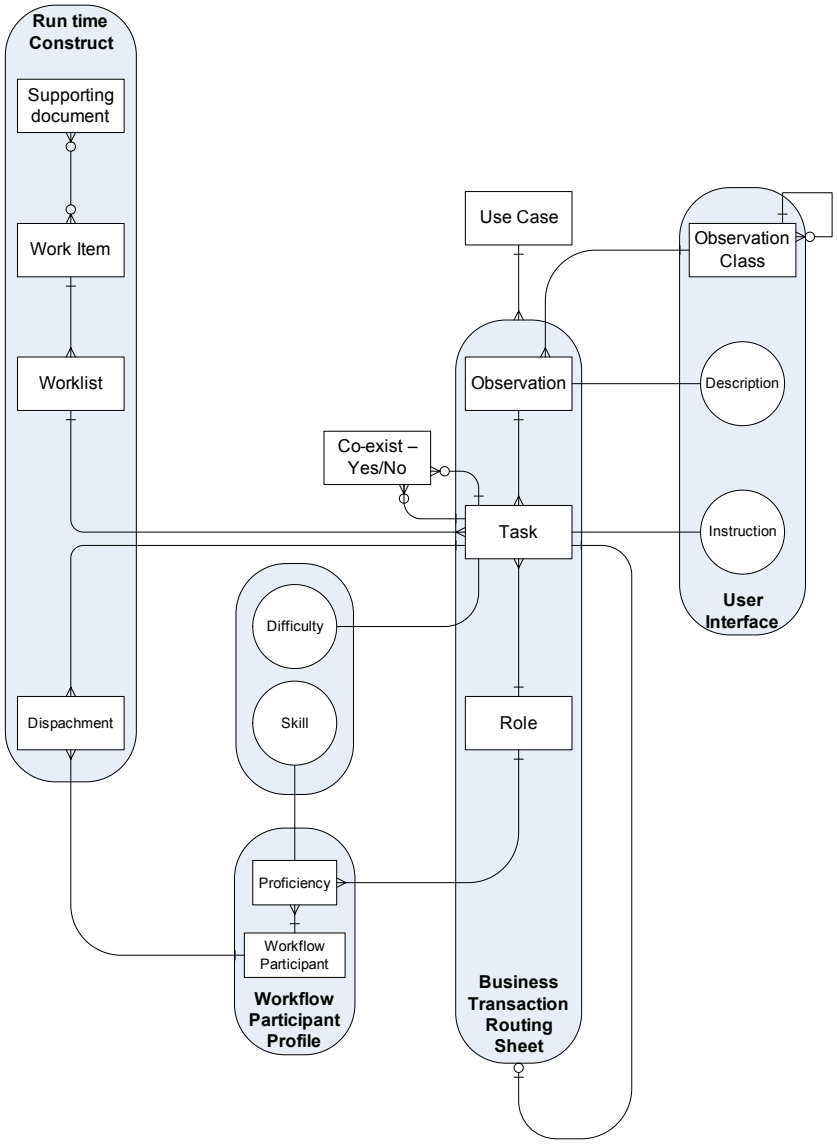


Figure 3.5 – Use case oriented workflow engine — Data Model

3.4.2 Methods — Run Time Interface

A use case oriented workflow engine exposes several methods at run time, enabling workflow participants to dynamically select the activities each workflow item is subjected to. The following definitions capture these methods.

Definition §3.10
observe, method. A public method used by a workflow participant to declare which use cases or use case extensions should be performed on a work item.

Definition §3.11
prescribe, method. A private method used by the workflow engine, and not exposed to users of the engine. In response to an observation, the workflow engine places activities in a queue according to the business transaction routing sheets. The activities queued are the action steps derived from use cases or extension.

Definition §3.12
GetNext, method. A public method used by a workflow participant to request a worklist from the workflow engine.

Definition §3.13
done, method. A public method used by the workflow participant to acknowledge the completion of an activity.

Definition §3.14
diarise, method. A public method used by the workflow participant to postpone the execution of an activity until a given date and time.

Definition §3.15

should not be done, method. A public method used by the workflow participant to instruct the workflow engine to cancel an activity already prescribed to a work item.

3.4.3 The Engine

Thus far, having started from basic principles and continued with commonly accepted workflow and requirement elicitation principals, we have laid the foundations for the introduction of the first contribution of our dissertation — the use case oriented workflow engine. We then described some properties of the use case oriented workflow engine. Now we will introduce our thesis — the proposition of a workflow engine whose workflow modelling language is the extended written use case language, following this we describe how a workflow engine configured with extended written use case languages is operated. The remaining chapters of this dissertation are devoted to the evaluation of the thesis.

Definition §3.16

use case oriented workflow engine, n. A workflow engine that accepts the extended written use case language as its workflow modelling language.

At run time a use case oriented workflow engine exposes the observe, GetNext, done, diarise, and should not be done methods. The dispatcher of a use case oriented workflow engine prescribes activities linked to observations. For after the-event-analysis, a use case oriented workflow engine posts messages to a business activity-monitor (*i*) when work items are created (*ii*) when work items are terminated, (*iii*) when activities are spawned, (*iv*) when activities are completed and (*v*) when observations are recorded.

In the next subsections we describe the activities needed to configure a use case oriented workflow engine, operate one and analyse the logs produced by such a workflow engine. We list the activities and group them into roles. We thus explain in detail the various roles required in an organisation to support a use case oriented workflow engine.

```
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
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:
       2a1. The Senior Data Entry Person also keys the
            deposit.                                 Order = 2
       2a2. The system reconciles the two data
            entries.                                 Order = 3
       2a3. The flow continues at line 3
            2a1a. The reconciliation failed
                  2a1a1. The system sends the two data
                         entries to the senior data
                         entry.                       Order = 4
                  2a1a2. Senior data entry corrects
                         the data                     Order = 5
                  2a2a3. The flow continues at line 3
2b. The form arrived unsigned:
    Replace action step 2 with:
    2b1. The Data Entry Person calls the
         Investor, requesting a signed form.  Order = 2
    2b2. The current process ends.            Order = 3
```

Figure 3.6 – The sample business process expressed as a extended written use case language (taken from [86]).

```
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 signature1
       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     =   Data Entry Person
           Order    = 5
```

Figure 3.7 – The business transaction routing sheet for our sample business process (Figure 3.6) (taken from [86]).

3.4.4 The Role of the Business Analyst

Elicit the Written Use Case Set

At configuration time, the business analyst, in cooperation with the representatives from the business, elicits the process that requires automation and articulates it as a use case set. This is a well-understood phase of many IT-based system life cycles, and use cases are a common language for the purpose.

3.4.5 The Role of the Workflow Configuration Officer

Elicit Order of Processing

The order of processing identifier cannot be automatically inferred from the use case. The workflow configuration officer has to clarify the issue with the business analyst. For example, consider Figure 3.6 on page 78. A reader familiar with the written use case language will notice a syntactical construct following each action step — `Order = n`. This is how we propose to describe the order of processing identifier of action steps.

Infer the Business Transaction Routing Sheets

At configuration time, the workflow configuration officer infers the business transaction routing sheets from the scenarios in the use case set. Every main success scenario, every alternate scenario and every extension in the use case set gave rise to a business transaction routing sheet. Every action step in each scenario gave rise to an activity in a business transaction routing sheet.

In example 3.7 on the previous page, the workflow configuration officer configured four individual business transaction routing sheets. The main success scenario gave rise to a business transaction routing sheet (see sheet 1 in Figure 3.7), each alternative scenario and each extension gave rise to a business transaction routing sheet and each extension gave rise to a business transaction routing sheet (see sheets 2,3 and 4 in Figure 3.7). They are all associated with observations in the individual business transaction routing sheets. Each group of action steps that follows them is the reoccurring activity element.

Infer the Roles

The roles of the workflow participants are found as the grammatical subjects of the action steps. To infer a role from an action step one has to identify the grammatical subject of a sentence. For example if an action step says that "5. The claims officer verifies the signature on page 15 of the application form", then the role of that activity is "claims officer".

Infer the User Interface Specification

The workflow configuration officer infers the observations and the instructions from the use cases. Observations are the use case titles and the conditions in the extension sections. The instruction is the predicate and the object in the action step. For example refer to Figure 2.8 on page 43 — The restricted grammatical structure of an action step. In that example we show how an action step can be parsed using syntactical analysis, and the transformation from syntactical constructs into the various use case oriented workflow engine terms. The workflow configuration officer transforms the subject of the action step into a role, and the predicate and the optional object into an activity. That is, in the action step *"The data entry person keys the deposit to the system"* " the phrase *"data entry person"* is the role, and the phrase *"keys the deposit to the system"* is the activity.

3.4.6 The Role of the Human Resource Team

Establish the User Profile

At configuration time, the details of the workflow participants 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 assume that the profile is populated by the human resources team.

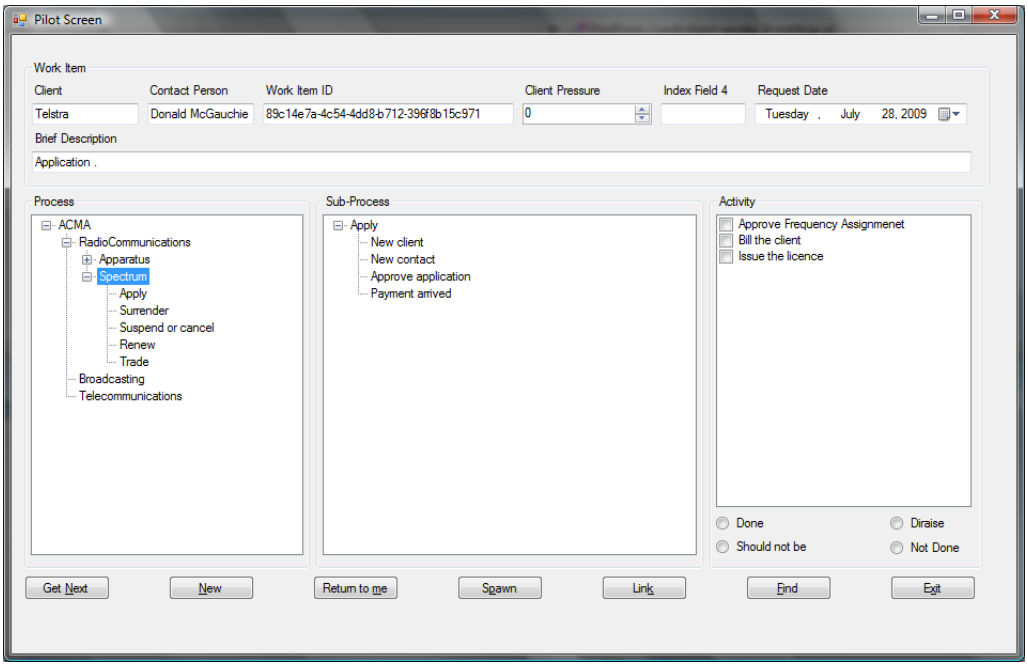


Figure 3.8 – An example of the run time user interface of a use case oriented workflow engine.

3.4.7 The Role of the Security Officer

Infer Minimal Rights

At configuration time, the security officer infers the minimal rights of a workflow participant from the written use case language as described by Fernandez *et al* [38]. This is done by identifying the action steps each role bearer performs and granting only the privileges required to do these action steps and not granting other privileges.

3.4.8 The Role of the Workflow Participant

At run time the workflow engine assigns worklists to workflow participants, and the workflow participants execute the work items in the worklist. In a clerical setting, the workflow participants invoke the following four methods:

Catalogue Workflow Items

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, as well as business specific dispatcher related information. For example, for an insurance claim, cataloguing may require the keying of policy ID, claim date and claim's estimated value.

Pilot

> **Definition §3.17**
> pilot, n. A person who steers or directs the course of a ship; a helmsman or navigator, spec. a qualified coastal navigator taken on board temporarily to steer a ship into or out of a port, through a channel, *etc* [25].

> **Definition §3.18**
> pilot, method. To prescribe or link one or many business transaction routing sheets to a work item.

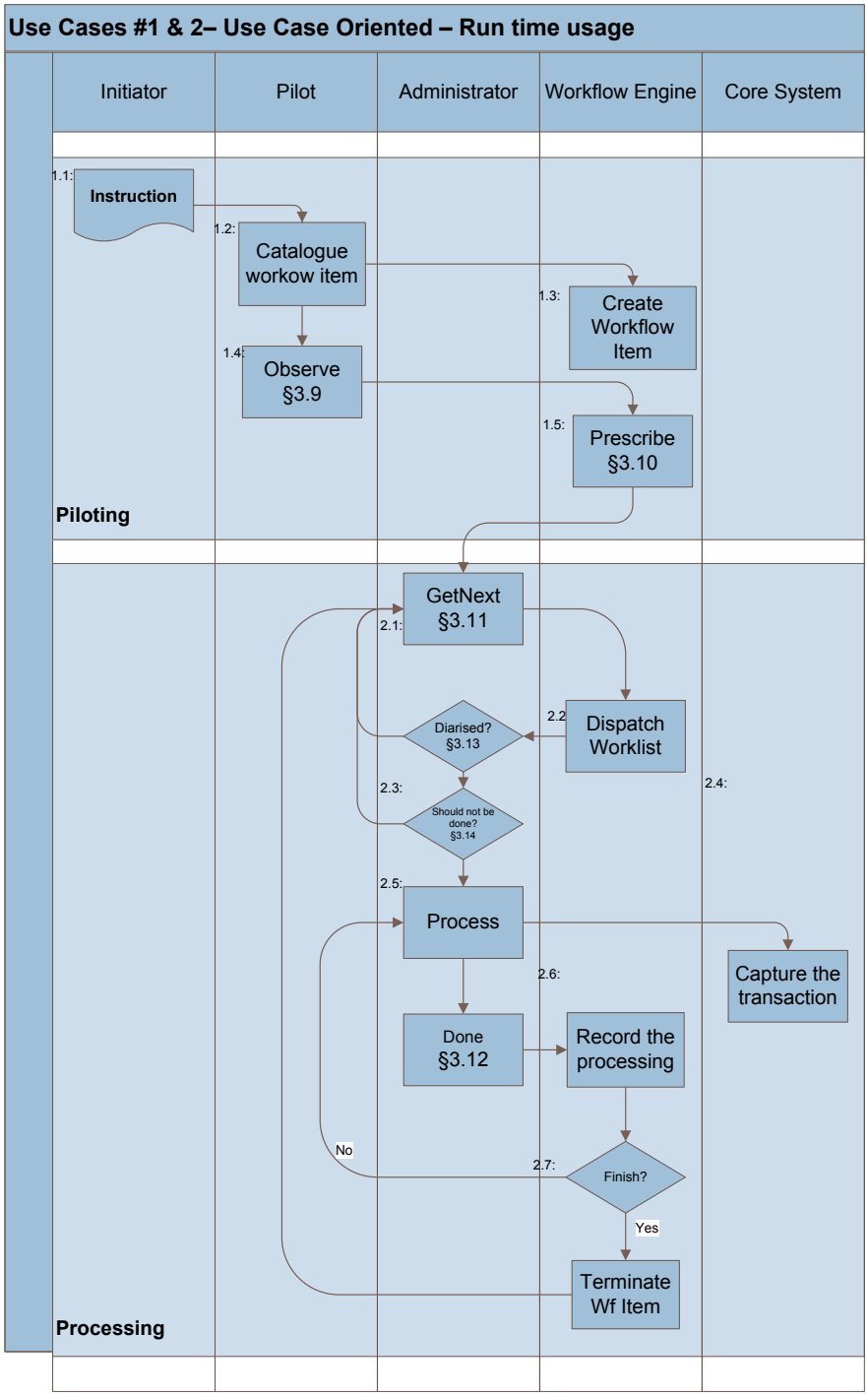


Figure 3.9 – Use case oriented workflow engine — run time usage. Here we describe two use cases — piloting and processing. Please note that the diagram has a formal flow: the business process never terminates.

As with a maritime pilot who describes the route of a vessel through a complex waterway, the pilots describe the route the work item takes through the organisational maze. The pilots do so by linking business transaction routing sheets to work item using a classified menu of observations. At this stage a work item 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, and in effect be a pilot.

As illustrated in Figure 3.8 the pilot first selects appropriate observations from a list presented on the left panel, in response the use case oriented workflow engine reciprocates by presenting all the sub processes (that is, extensions) of the selected process, in the middle panel. The pilot then selects all the sub processes that are applicable to the work item. Making these selections is called "piloting". In due course the workflow engine presents the work item to an appropriate workflow participant in the right panel, who in turn performs the activities (read action steps) and acknowledges the performance. On some occasion, the workflow participant may postpone (diarise) the execution of an activity or declare that it "should not be done".

Execute

To prevent double handling, the pilot may execute the activities there and then, or leave the activities to a specialised workflow participant. At this stage, the workflow participant requests the next work item by calling the workflow engine's "GetNext" method, and the dispatcher dispatches a work item (taking into consideration the cataloguing attributes that were previously assigned, the supporting document(s) and, according to the observations the pilot had prescribed, a list of activities the workflow participant is expected to perform). The workflow participant then performs the appropriate activity on the work item. For example, in figure 3.6 on page 78, the activities are "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 also flags other activities as "diarise" or "should not be done", until all activities are dealt with. Following the acknowledgement, the workflow participant may either request the next work item or terminate the session.

3.4.9 The Role of the Dispatcher

Infer Worklists

At run time, the workflow engine groups one or more activities into a worklist and dispatches the worklist to a workflow participant.

The dispatcher watches two lists:

- available workflow participants with their roles; and
- piloted activities with roles.

These two lists are used to implement dispatching patterns that the business process managers define. Examples of dispatching algorithms used in BT Financial group are provided in section 4.4.4 on page 100.

Populate the Business Activity Monitor

At run time the workflow engine sends messages from the following three classes to the business activity monitor:

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. Used to monitor adherence to external Service Level Agreements.

Observation Message: Sent when a business transaction routing sheet is assigned to awork item. Records properties of a work item. Used for quality assurance (*e.g* in Figure 3.6 on page 78, how often do investors forget to sign application forms).

Activity Message: Sent when an activity is queued, starts, ends, or diarised. Records who prescribed the activity, who performed the activity and how fast. Used to monitor adherence to internal service level agreements.

3.4.10 The Role of the Business Process Manager

Define the Dispatching Algorithm

At configuration time the business process manager selects the dispatching algorithem used by the use case oriented workflow engine. The number of the dispatching algorithms is limited only by imagination. We list a few of the dispatching algorithms tried in BT Financial Group in section 4.6.

Review the Audit Data

After the event, the business process manager reviews the audit logs and finds processes that take too long and processes that generate too many defects. Following a correction intervention, the business process manager reviews the logs again, finds if the intervention worked as expected, and continuously improves that system. Another of the major challenges of the business process manager is to learn lessons from the logs when designing the work flow participants' rosters.

3.5 Conclusion

When one maps syntactic constructs to semantic human endeavours, syntactic certainties vanish. A concrete example: the syntactic meaning of a BPMN rectangle is in agreement, but the semantic meaning of the label inside a rectangle is open to interpretations like any other natural language statement, albeit stylised.

Workflow configuration officers can map workflows modelled using the extended written use case language to BPMN diagrams. The written use case language's action steps, extensions and stakeholders give rise to BPMN's rectengels, diamonds and swimlanes respectivly. The extended written use

case language's order of processing identifiers give rise to the arrows in the BPMN diagrams, these arrows then connect rectangles to diamonds, rectangles to rectangles, diamond to diamonds and rectangles to diamonds. BPMN's explicit fork join symbols, as well as start and end symbols are missing from Cockburn's notation, but are implicitly present none the less. The extended written use case language trigger structure is translated into BPMN start event, the end of the main success scenario is translated into BPMN end event. As well the end of an extended written use case language extension is translated into BPMN fork join symbol.

The written use case language is formal. The written use case language has a clear syntax, and errors can be identified, not only for the "subject – verb – object" syntax of action step. For example, questioning the preconditions of a written use case language is forbidden within the written use case language.

Cockburn focus our attention on semantic meaning, rather than on the syntactical frame within which the semantics are articulated.

Thus, in this section we have demonstrated that a workflow engine whose workflow modelling language is an extended written use case language can be built configured, used and monitored.

3.6 Reflection

Activity theory provided me with the theoretical foundation required to emphasise the need for sustained dialogue between the workflow configuration officers, pilots and workflow participants, rather than the command model assumed by the somewhat more rigid contemporary workflow management approaches. Indeed, with my proposed use case oriented workflow engine, pilots regard the generic workflow models as a starting point for the design of the specific handling of work items and not as a process that detached workflow configuration officers force upon them. I devoted a couple of pages to activity theory in section 2.1.5 on page 32 — Tuning, and to activity psychology in section 2.3 — Computer Human Interaction.

I observed the business-process modelling process in several organisations, and analysed the communication between workflow participants, busi-

ness analysts and workflow configuration officers. Taking in mind the dynamic nature of process modelling, I introduced a workflow modelling language using syntax I borrowed from the requirement elicitation domain — the written use case language.

I described how does a use case oriented workflow engine asks certain workflow participants to link goals with a work item, and how does the workflow engine reciprocates by selecting the activities that workflow participants need to perform. Using my proposed use case oriented workflow engine approach, where everybody is a pilot, workflow participants are empowered to accept the routing prescribed by pilots with a grain of salt. With a use case oriented workflow engine, workflow participants can prescribe activities ignored by the pilots, mark activities as 'should not be done' or diarise activities.

The requirement engineering community gave me a detailed syntactical reference model of two use case languages. Jacobson described the concept and Cockburn described the written use case language, a very expressive use case language. The written use case language is more expressive than Jacobson's UML use case language, as only the former has explicit "OR split" and "OR join" building blocks building block.

I placed the term dynamic in the title of the dissertation. In contrast with incumbent workflow engines that require a full release cycle to propagate any change, a use case oriented workflow engine is dynamic, changes to workflow definitions can be done at every stage of the process without a need to flush the system.

Changes can be introduced by business analysts, workflow configuration officers, pilots and workflow participants. Changes can be done at two levels: (*i*) the workflow process in general, usually when new activities are discovered, and (*ii*) at the specific work item when better informed workflow participants decide to override a pilot's earlier recommendation. With a use case oriented workflow engine, workflow participants regard the workflow model as a tool that helps them to perform their work. Pilots use the prescription module to tailor the activities that will be performed on each work item. They use the workflow model to guide them, but are empowered to change it if needed. Workflow participants on the other hand

are not obliged to fulfil the pilots' instructions. The workflow participants can always mark some activities as should not be done. They are empowered to postpone activities by diarising them, as well as prescribe extra activities as they deem right. Some workflow participants are even empowered to create new activities that will be used by other workflow participants later. As all of this is audited in detail, moral hazard is reduced.

When analysing rich context [34, Section 18.3] which is closer to my domain then the usability of a simple GUI, Dix *et al* help me establish the theoretical foundation for my method by stating "when we have a goal / sub goal model, we can create internal plans which are blindly executed" [34, Page 642]. This mode of operation similar to the observation / activities model I later promote. In many ways this section, with its focus on goals, tasks and triggers [34, Section 18.3.3] is very close to Cockburn's approach [24]. Dix *et al* state that there is no reason why most task analysis methods should not adopt some form of artifact tracking where the authors clearly describe a workflow engine [34, Page 647].

That decomposition is very applicable to my work: — the pilot's role, I later propose, concentrates on the goal identification and operation prescription, the workflow participants concentrate on execution and the workflow engine on completion and wrapping up tasks. My proposition is that different populations of workflow participants are skilled for operation selection compared to those skilled for operation application. In BT I found that for complex operations, the selection is sometimes assigned to less trained workflow participants while for simple, yet varied operations, the selection is sometimes assigned to a highly skilled team member.

This can all be done if the workflow modelling language is expressive and if the workflow modelling language is readable by a wide cohort of stakeholders. I test the former statement in chapter 5 – Expressive Power. I test the latter a statement in chapter 6 — Readability. All together, I refer to this approach as a *dynamic workflow engine*.

In the remaining chapters of this dissertation we will see how a workflow engine of such style was accepted by the industry, how expressive is the language (in comparison with other languages, and evaluated by pattern library), and how readable is the language.

Chapter 4

Case Study

> and it was written in the book of the chronicles
>
> Esther 2:23

BT, then Bankers Trust Australia Limited, a fully owned subsidiary of Bankers Trust Company of New York, and now BT Financial Group, the financial services arm of Westpac Banking Corporation, built a use case oriented workflow engine. The author of this dissertation was involved in the implementation of that use case oriented workflow engine as a programme manager.

Here we study what happened at BT. We describe the system's environment, concentrating on the business problems that BT confronted. We describe the evolution of the business process management practice within BT and discuss how the implementation of a use case oriented workflow engine worked out. We conclude by summarising the benefits BT Financial Group derived from its use case oriented workflow engine.

4.1 Methodology

The methodology we employ in this chapter is the detailed study of a single-case, a methodology that was popularised by the Harvard Business School.

A strength of the methodology is that a detailed study of a single-case develops a nuanced view of reality, a level of detail that is needed as the human behaviour cannot be meaningfully understood as governed acts. A detailed case study is also a fertile ground for forming hypotheses, as indeed was the case in this research programme. Description of concrete, context-dependent knowledge is, therefore, a valuable part of a scientific work [42, Page 222-5].

A weakness of the case study methodology is that a single-case study cannot be used to arrive at a general conclusion that would apply in all cases. Thus this chapter can be seen as informative rather than definitive.

4.2 Background

BT Financial Group, now a fully owned subsidiary of Westpac Banking Corporation, with roots in the deregulation of the financial markets in Australia in the 1980s, is an Australian company that offers a comprehensive suite of wealth management products including life insurance, managed funds*, superannuation†, cash management as well as a Wrap platform that administers portfolios managed by sophisticated investors‡. These wealth management products are governed by multiple frameworks of rules: legislative, commercial, operational, audit controls and revenue assurance.

For example, some rules are set by the Australian Taxation Office regarding eligibility to invest, say in retirement products (e.g. only Australian residents can deposit funds into a superannuation product). Commercial rules regard amounts that can be invested (say minimum balance of AUD$2,000$ has to be maintained at all time for the investment to bear interest), cut off dates and time (say instruction arrived by 10 a.m. will be processed

*Managed funds are called mutual funds in the US.

†The Australian superannuation system is somewhat equivalent to $401k$ framework in the US.

‡The Australian Wrap platforms offer services that can be compared to the fully serviced brokerage accounts in the US

on the same day) and more. Operational rules govern the internal service level agreement, the data entry processes, error correction protocols, quality control activities that are performed by various workflow participants (say who should type a transaction, who should retype parts of the same transaction and how should a discrepancy between the two be handled), and more. Audit controls force, for example, the separation of back office from front office. Revenue assurance verify that the group is collecting all the fees it earned.

The complexity of these ever evolving business rules made it hard to computerise all of these rules using one or even several mainframe systems. While BT aimed at a high level of computerisation by the main data processing systems, some business rules necessitate manual intervention, often due to the developers not being able to cope with the annual cycle of legislative changes, as well as new business rules devised by the marketing department, that at times reacted to offerings of competing organisations. This fragmentation gave rise to rules maintained in spreadsheets, maintained in dedicated online calculators, stored only in the head of administrators and recorded in legal documents inaccessible to anyone but the legally trained. Indeed it was not uncommon for dispute to arise between departments within the group and between the group and its customers. The resolution of some of these disputes were so complex that legal and actuarial teams had to be involved in their resolution (think of correcting a transaction recorded in a previous financial year, for which the books are closed, tax paid and income distributed). Consequently a complex web of departments and roles was established. Indeed the group had more than 500 distinct roles in place in 2006.

The complexity of the management of a web of business rules necessitate the breaking of processes into a set of activities, each earmarked to a dedicated pool of role bearers. The challenges that BT's management faced were the traditional business process management challenges we described before, namely the modelling, verification, enactment, monitoring, and tuning of business processes, as well as enforcing and verifying compliance with the above mentioned business rules.

To address these challenges BT introduced two generations of workflow

engines, Basil and its successor BTeP.

4.3 Basil

Between 1993 and 2001, BT established Basil, a system that leveraged FileNet's imaging technology. The metaphor behind Basil was that of an office environment where administrators moved documents from a virtual *in-tray* to a virtual *processing-tray* and then to other *processing-tray*s until such time as an administrator completed the processing of the document and archived the document in a virtual *file-room*.

Following this, between 2001 and 2005, using six tactical projects and one strategic project, BT gradually migrated its business process management practice from a tray based imaging system to a use case oriented workflow engine. Each project was strictly controlled and cost justified. All projects had a return on investment of less than two years.

While using Basil, on a typical day, BT scanned 12,000 documents. The operators of the scanners roughly sorted the documents into several pigeonholes and then scanned the documents from each pigeonhole into a corresponding Basil *in-tray*. Dedicated senior administrators monitored each Basil *in-tray*. Upon arrival of a supporting document into a Basil *in-tray* the dedicated senior administrator pulled the supporting document from the Basil *in-tray*, indexed the supporting document, and annotated the supporting document by typing free text onto Basil's supporting document annotation structure. That senior administrator then pushed the supporting document to an appropriate Basil *processing-tray*. Dedicated administrators monitored each Basil *processing-tray*. Upon arrival of a supporting document into a Basil *processing-tray* an administrator who monitored the *Basil processing-tray*:

1. pulled the supporting document from that tray and implicitly locked the document

2. read the annotations

3. processed the supporting document as per the annotations

4. added their own annotation

5. pushed the supporting document to another Basil *processing-tray* for further processing by other administrators and implicitly unlock the document

This process was repeated by other administrators until the processing of the supporting document was completed. The last administrator placed the supporting document in the final Basil tray - named *file-room*.

This practice left BT exposed at the following fronts:

- The monitoring of large number of Basil trays and the assignment of workflow participants to appropriate trays consumed a significant amount of management time.
- The imaging system did not provide mechanism to decide to which of the thousands of trays a document should move after every processing stage. It was the administrators' job to know the dozen or so Basil *processing-trays* that were relevant to their business process and they had to push supporting documents to these *Basil processing-trays* diligently.
- Processes had to be put in place to locate work items that were misfiled.
- Excessive typing of annotations caused repetitive stress injury (RSI)
- Lack of standardisation of the annotations disabled automated routing
- Lack of standardisation of the annotations disabled-after-the-event analysis. To resolve that issue artificial Basil trays (and tray movements) were needed to record attributes of workflow items.
- Free text was subject to ambiguities
- The annotations were not always up to BT's writing standards. With the emergence of freedom of information, clients were entitled to request access to their files, exposing BT to reputation risk in the process.

To address these challenges, BT commissioned the *comment-assistant*, a tiny Visual-Basic annotation prescription system that leveraged FileNet imaging programmable interface. That tiny utility, as we explain later, was the little acorn from which the use case oriented workflow engine grew. The *comment-assistant's* configuration file was made of records. Each record had four fields: (*i*) an observation, (*ii*) an activity, (*iii*) Basil *processing-tray* – roles and (*iv*) order of processing identifier. Eg:

- ‘‘Signature is missing” — observation
- “Client service administrator to contact the investor requesting a new form” —activity .
- “Client-Contact-Tray” — role,
- “2” — order of processing identifier.

To initialise the *comment-assistant*, business-analysts or line-managers populated the *comment-assistant's* configuration file. At run-time, the senior administrators who monitored that Basil *in-tray* pulled the supporting document and selected appropriate annotations from the *comment-assistant*. The *comment-assistant* in turn pasted the annotations into the document and pushed the supporting document to a Basil processing-tray associated with the observation. Following this, administrators who monitored these Basil *processing-trays*

1. pulled the supporting document,
2. read the annotations
3. acted accordingly, and
4. placed their own annotation on the document
5. .. until the completion of the processing where the comment-assistant pushed the supporting document to the Basil *file-room*.

Having fed a few scores of business processes into the comment assistant, business analysts, including the author of this dissertation, had a road to Damascus realisation: the Cockburn style use cases, which business analysts

write in order to specify a process, could ease the identification of the annotations. The only addition BA's in BT introduced to the written use case language was to add explicit support for parallel steps – order of processing identifier as described in section 3.3 — Extending the Input Language. BT had to include partial order as it was a fundamental business requirement.

The deployment of the *comment-assistant* was followed by a series of tactical improvements to the imaging system, such as adding the ability to initiate a workflow from a Microsoft-Office application (Save to Basil) and ability to spawn work items from existing work items (Basil Photocopier). All these extensions still retained the fundamental constraints of the imaging system namely:

- a significant part of the middle management time was consumed by the inspection of the content of Basil trays and the assignment of workflow participants to appropriate trays (a daily ritual dubbed by some larrikins[§] as *"horse trading"*),
- inflexible ratio of one supporting document to one work item,
- dependency on the clerical accuracy of workflow participants for the determination of routing trays,
- the administrators had to perform redundant tray movements in order to record attributes of workflow items.
- Cherry picking. This refers to the action or process of selecting only the best or the most profitable items, opportunities [25] – Some processors picked work items according to their needs rather than the customers' needs.

4.4 BTeP

To resolve the fundamental issues intrinsic to an imaging system, a strategic shift was suggested – the adoption of a workflow engine. BT conducted a formal evaluation of the market using a request for proposal process and

[§]Larrikin, n. A (usually juvenile) street rowdy; the Australian equivalent of the hoodlum or hooligan [25].

chose a workflow engine supplied by FileNet, then an independent company and now a part of IBM. While a few the competing workflow engines were of equal functionality, BT hoped that an integration partner FileNet introduced would reduce the implementation project duration by six month.

A fundamental feature of FileNet's out of the box solution was that for each use case, a dedicated BPMN style diagram was expected. That was deemed to limit the scalability of the system as it required too much development resources. BT had more than 350 use cases in mind and empowerment of front line supervisors was thus required. These supervisors were more likely to have an accounting related degree than an IT related degree. BT have thus introduced the use case oriented workflow engine to the configuration of BT's FileNet basedworkflow engine.

BT built BTeP, a use case oriented workflow engine, as an application that called the FileNet workflow engine application programmable interface according to the framework described in Chapter 3. The first use case went into production after six months of development, and the second took yet another six month, which were devoted to generalise BTeP to support multiple use cases. From then on, business processes were automated at the rate of one per month, accelerating eventually to the rate of one per week.

The high quality of data, the accurate models of business processes, the efficient assignment of roles to workflow participants and the accurate indexing of work items, all presented BT with the opportunity to programmaticly enact and monitor its workflow. This section describes further refinements to the use case oriented workflow engine, which business-analysts in BT Financial Group found to be useful.

4.4.1 Skills and Difficulty

The initial configuration of BTeP saw the assignment of roles to workflow participants according to the use cases in which the workflow participant could participate as actor. Roles where thus, for example "Retirement products redemption officer", "Cash deposit officer", or "Dispute resolution office". Very soon, it became apparent that an increase in the granularity of the roles catalogue was required as some of these extensions were easier than other. For example, some switches between superannuation products were

harder to process than switches between managed fund products. Close analysis found that, using written use case language terminology, all the workflow participants whose role permitted them to process switches between superannuation products could handle the main success scenario, the 'happy day scenario', but not all workflow participants could process all of the extensions. In this case, some extensions required calculations that had to be processed using complex spreadsheets.

To increase the granularity of the dispatching of activities to role bearers, BT's business analysts assigned difficulty to activities and skill to role bearers. BTeP's development team configured the dispatcher to assign worklists to role bearers who were sufficiently skilled to handle the most difficult activity in the worklist (more about the dispatcher later). This enabled the introduction of seniority amongst role bearers, difficulty amongst activities, and the appropriate despatching of achievable activities to administrators according to their seniority and the complexity of the activity.

4.4.2 Observation Menu

To increase operational efficiency and increase the speed of processing mail, BT Financial Group outsourced the scanning of supporting documents to Australia Post, shaving a full business day from the time elapsed between the moment a customer posted an instruction to BT and the time the instruction was processed. The central scanning facility implied that all the work items arrived to one pool where they were roughly sorted, but further sorting capability was required. That secondary sorting required detailed understanding of product rules.

Close analysis found that, using written use case language terminology, an efficient way to catalogue work items was by business line (say Superannuation, Cash, Insurance...), then by use case title (say application, redemption, switch, transfer...), and then, by observations. These observations, as we have described before, linked work to business transaction routing sheets. All that the pilots had to do was to prescribe or link one or many business transaction routing sheets to each work item.

To ease the task of locating observations, a taxonomy based tree structure was implemented. This fitted very well within Newell and Simon's sug-

gestion that problem solving exercise has four steps: goal forming, operation selection, operation application and goal completion. The observation menu helped the workflow participants to separate operation selection from operation application, either by separation of duty between pilot and processors, or by separation of state of mind, if the pilot was also a processor [34, page 422].

4.4.3 Index

To catalogue work items, BT used a key combining the customer's identifier, the business line and the transaction type. Supporting documents were catalogued in BT Financial Group using monotonically increasing, non contiguous integers, with a check digit concatenated. XML documents would be usually catalogued by a computer programme reading the document and assigning attributes to the indexing of the document according to the content of the document. Every business line had a different indexing strategy, which at times was expressed as varying number of fields, a fact that created a programmatic challenge to the GUI developers.

4.4.4 Dispatching

Dispatching is one of the fundamental reasons to have a workflow engine. The dispatcher decides what is the order of the workflow items in general, and amongst the sorted list of workflow items, which is the one that a workflow participant would receive the once workflow participant invokes the GetNext method. Naturally, when designing the dispatching algorithm one has to take into consideration diverse elements such as workflow participants skills set, the rarity of a specific skill the workflow participant possesses, internal and external service level agreements, fairness and more. In BT, different business lines had different dispatching algorithms at different times of the day and at different times of the month, taking into account daily processing cycles as well as annual processing cycles. The dispatcher took into consideration data elements such as: value date, product, client pressure, value, and distribution channel.

BT experimented with the following dispatcher patterns:

- FIFO. This mode which seems the fairest was often found to be the least efficient, as it can overwhelm skilled workflow participants with mundane tasks.

- The hardest job one can do in descending age order. This mode of operation seems to better utilise highly skilled workflow participants, but not to be fair.

- The oldest un–piloted work, then the oldest and the hardest work item a workflow participant may perform, from the oldest day. This moderately complex dispatching rule seems to be fair and efficient.

The basic dispatching rules were sometimes further modified, taking into account extra information. For example the dispatcher considered:

- Business related consideration such as priority for redemptions over deposits and of cash transactions over manged fund transactions. This rule tries to help an organisation with competing demands to handle the most valuable of them.

- Client pressure, an integer that enables manual control the priority of a work item. This pragmatic adjustment was needed to let the workflow participants feel in full control of the system's behaviour at the price of efficiency and fairness.

4.4.5 Roster

The meticulous documentation of the arrival patterns of work items on one hand, and of the productivity of workflow participants on the other hand, enabled BT to model its staffing levels using an Erlang based rostering engine.

4.4.6 Workflow Patterns

BTeP's development team identified several workflow patterns which required the workflow participants to prescribe a sequence of activities, some

of which required some optimisation of the dispatcher. These workflow patterns were needed to support very specific business rules that were grounded in the organisation's values. For example:

- The quest for excellence in client service, necessitated that whenever possible, the same client contact representative will contact the customer through the entire life of the work item.
- Another example is that to increase data quality, some data items had to be keyed by two different workflow participants.

Some of the patterns may not be common to other organisations due to difference in business domain and in culture. BTeP's development team found it worthwhile to hard-code the patterns into the use case oriented workflow engine, rather than to ask workflow participants to prescribe them individually. We now list several of these patterns.

Boomerang

The Boomerang pattern, or 'return to me', describes the movement of a work item from one workflow participant (owner), to a second workflow participant and then back to the first. The pattern emerged because respect for the individual, a cultural value, required that a work item will return to the same workflow participant if further processing is required by an identical role bearer.

For example if a transaction required some evaluation by one workflow participant, an approval by a second workflow participant, and then execution by yet another workflow participant of equal role to the first workflow participant, then it made sense to assign the third activity, whenever possible, to the first workflow participant who was familiar with the work item. It would have taken up to twenty minutes to establish this familiarity, had the dispatcher chosen a different workflow participant.

A specific example is an application for life insurance policy, that requires processing by a fund administrator, medical evaluation by an underwriter, and then further calculations done by a fund administrator. Assigning the first and third activities to the same workflow participant would significantly expedite the third activity which is further calculations.

A complexity inherent to this pattern is that the BTeP's development team had to put a process in place should the work item owner be hit by the proverbial bus, or more generally, when the first workflow participant is unavailable for a protracted period of time (defined by the service level agreement), in that case the use case oriented workflow engine has to assign the third activity to a third workflow participant.

Implementation of this pattern using the normal GetNext method was not possible as workflow participants were not allowed to explicitly select work items due to our wish to prevent 'cherry picking'.

Verification

Verification is a pattern that moves the work item to a different workflow participant with an identical role for independent review.

The verification pattern emerged, because quest for quality, a cultural value, required that certain data elements were typed twice by two independent administrators. For example, during the processing of an application into a managed fund, the investor's bank account was keyed by two workflow participants of equal roles.

As for boomerang, implementation of this pattern using the normal GetNext method was not possible as workflow participants were not allowed to explicitly select work items due to 'cherry picking' prevention.

Call planner

In this pattern the use case oriented workflow engine presents a batch of several related work items (worklist). For example, if a client contact is required, the use case oriented workflow engine should make sure to address all open issues with the customer, even if many of them are of a lower priority than work items at hand. This pattern reflects the need to cover as many issues as possible when establishing a contact with a customer.

Diarise

Workflow participants could place the work item on timed hibernation before despatching another activity. For example, when contacting a cus-

tomer, and learning that the customer is on protracted leave, the workflow participant may place the work item on hold until a day or two after the client's leave.

The normal GetNext had to be tweaked to achieve this pattern as the priority of the work item would have otherwise increased and the GetNext algorithm would have assigned the same work item whenever invoked, like a broken record.

Floodgate

The Floodgate pattern which is a variation on the diarise pattern, requires the GetNext method to wait for an external event and then move to another activity. For example, selling instructions have to wait until a price is set. When a price is set, a floodgate opens and the dispatcher despatches work items that depend on that price.

Just as above the normal GetNext behaviour had to be changed as the priority of the work item would have otherwise increased and the GetNext algorithm would have assigned the same work item whenever invoked, like a broken record.

The Floodgate pattern may have been too complex to explain, and while deployed and maintained, it was never used.

Cluster

The Cluster pattern enable pilots to prescribe several activities in one mouse click. The pattern is useful when several activities are often prescribed as one.

4.5 Implementation on Off The Shelf Engine

While BTeP's development team could have implemented a use case oriented workflow engine on a great variety of platforms, the team elected to implement a use case oriented workflow engine on top of Panagon WorkFlo *[sic]*, a commercial workflow engine from FileNet which was at the time an independent company and is now a division within IBM.

A challenge that BTeP's development team faced was the sheer magnitude of the problem at hand, namely the more than 360 constantly evolving distinct business processes (with an average of 26 activities each) that required automation. BTeP's development team could not afford the traditional workflow configuration practices as the practices required more developers than BT could hire, train or brief. BTeP's development team thus opted to implement and maintain the automation of this pool of processes by delegating responsibilities and authority to the workflow participants themselves.

These workflow participants, whose background was most often in the accounting trade rather than in software development, found it hard to model the extension rich business processes using FileNet's proprietary workflow modelling language which is a flowcharting based notation. As well, the workflow participants found it hard to comply with the software delivery life cycle best practice in general, a particular difficulty was source code management. To support and train these workflow participants, BT created a small change management team that trained the workflow participants in modelling their processes using the extended written use case language and in using BTeP.

As the workflow participants evolved the process constantly, often changing it several times a week, the business users could not afford to flush out all of the workflows in the system before releasing a new one.

BT's workflow participants were quite skilled with data entry in general and with browser-based data-entry in particular. It was thus natural to leverage these skills and develop a table driven user interface instead of FileNet's existing graphical interface. BTeP's development team utilised the form management capabilities of the FileNet platform to create five major screens, (*i*) for the capturing the extended written use case language, (*ii*) for maintenance workflow participants' roles, (*iii*) for indexing piloting and processing of work items,(*iv*) for enquiries and reporting and (*v*) for the simulation of various dispatching algorithms.

BTeP's development team used Microsoft SQLServer (called from IIS) to manage records such as workflow participants' roles and skills, the tabular workflow configuration and the business activity monitor.

The FileNet platform served BT very well, as BT was able to leverage it for the management of the user interface, the workflow participant's credentials, document management, queues, transaction integrity management and security log.

As BT Financial Group's GetNext method governed the despatching of work items from the moment they were spawned to the moment they were completed, BTeP's development team decided to configure the fundamental business process using the 'hub-and-spoke' topology (a topology that is also known as 'spider workflow' [20] or 'cross connected star').

With its 'hub-and-spoke' topology, BT had a central hub, that implemented the GetNext method, and a number of 'spoke' steps connected to the hub using bi-directional arrows. The GetNext step, determines to which activity (or activities) the work item should route to next. During the workflow run time, the work item moved from one spoke (workflow participant) to another, until a specific condition was met such that the workflow was completed. In this manner, any spoke can follow a previous one (even the same spoke more than once if required).

This pattern was especially useful as the order of activities could not always be determined at design time. Dynamic adjustment was needed when several steps had to occur more than once (for example the iteration of multiple proposals) or when workflow participants determine the next step of a process (for example when a workflow participant requires clarification from the customer). The 'hub-and-spoke' topology enabled the dispatcher to move work items from one arbitrary workflow participant to another arbitrary workflow participant, depending on the properties of the specific work item and use case on hand. This cross connected graph was the only workflow model that the FileNet workflow engine was conscious of. Above this layer BT implemented a use case oriented semantic layer.

BTeP's development team configured the FileNet workflow engine to have only four locations where work items would reside. (*i*) The unpiloted work queue, as its name suggests, held all work items that were not catalogued. As cataloguing all incoming document was a priority, this queue had a higher priority than the work in progress queue. (*ii*) The work in progress queue was the queue from which the dispatcher allocated activi-

ties to workflow participant. (*iii*) The diaries queue contained all the work items that were postponed — placed in the diary . (*iv*) The completed work store (named fileroom) was used for the storage of completed work items that ended.

4.6 Experience Gained

In this section we list observations of the system as it executed in practice:

Volumes

- In April 2007, 368 Business-Process were controlled by the system. By March 2011 this number increased threefold to 1,109.
- In April 2007, each business process had on average 26 possible activities. By March 2011 this number decreased to 20.
- In April 2007, on a typical day, about 600 workflow participants (administrators) were logged in. By March 2011 the number of workflow participants logged in to the system varied between 800 to 900.
- On an average day in April 2007, approximately 10,000 work items (business process instances) which were supported by 12,000 supporting documents (images) were executed. By March 2011 the number of supporting documents scanned daily increased sixfold to 75,000 but the number of work items (business process instances) remained identical. The ratio of supporting documents changed over this period.
- The number of audit rows generated daily was about 300,000. That number stayed the same in March 2011.
- The workflow participants (administrators) were located in three Australian states and in India.

Acceptance of the Configuration Mode

- The introduction of the GetNext method reduced the need for management to inspection the content of queues and the assignment of

workflow participants to appropriate queues. For a few years after the introduction of Implementation on Off The Shelf Engine management continued to inspect the content of queues and manually assign roles to workflow participants. As confidence in the GetNext method increased, a cultural shift occurred, and management learnt to accept the recommendations of the use case oriented workflow engine.

- The extended written use case language became the primary process modelling notation used by the workflow configuration officers.
- Workflow configuration officers' productivity was so high that in 2006 Westpac Life, a sister company which, as its name suggests, is in the life insurance business, migrated its entire processes into BT Financial Group's workflow engine within five weeks. For this achievement FileNet awarded BTeP's development team the best return on investment award for 2006.
- Line managers, with general accounting skills, felt comfortable to add, maintain, or remove action-items.
- Assigning the order of processing to action steps was never difficult.
- We found that the large majority of business processes did not have any scope for within-instance parallelism.

Business Activity Monitor

The workflow engine used event-driven asynchronous messages to communicate with an external data warehouse — the Business Activity Monitor.

Analysis of the audits data stored in the business activity monitor was instrumental for the identification, quantitative justification, and subsequent quantitative evaluation, of Six-Sigma process improvement initiatives.

Timing

New customers details were keyed into the systems as part of the piloting activity.

The dispatcher created worklists whenever a workflow participant requested the next worklist. This approach, where administrators request a worklist (GetNext) and the dispatcher assigns them the most appropriate one, rather then letting administrators "Cherry Pick" work items, increased the management control.

Piloting Strategies

BTeP's development team experimented with the following piloting patterns:

- 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 in an interleaved fashion throughout the day.

4.7 Benefits Found

Our experience showed that the use of a workflow engine configured by use cases gave the following benefits, which could be summarised for managers as the value proposition of our approach. BTeP resolved some of the issues that we highlighted earlier. In particular:

- The choice of the next routing step for a workflow item to follow, moved from the memory of an administrator to the workflow engine, that relied upon observation and their related worklists.

- As the typing of annotation using a keyboard was replaced by the selecting of tasks using a pointing device, typing was reduced, and the risk of RSI reduced.

- The standardisation of observations enabled the automatic prescription of activity to workflow items.

- As well, standardisation enabled the analysis of logs, eliminating reliance on artificial tray movements in the process.

- The standardisation of observations and activities enabled the organisation to invest the effort of eliminating ambiguities in the syntax of the observations and the activities .

- Dependency on the GetNext method removed the ability to cherry pick. If the workflow participant pressed GetNext, and received an undesirable activity, pressing GetNext again would present the same workflow item, until such time as another workflow participant pressed GetNext and received that workflow item.

- As well, standardisation of observation and activities enabled the organisation to ensure that no offensive language or offensive concepts were used in internal communication.

BT's experience showed that the use of a workflow engine configured by use cases gave the benefits which could be summarised for managers as the value proposition of our approach:

- A use case oriented workflow engine 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 were maintained as part and parcel of the process modelling reducing development effort. In a post deployment interview, management in BT Financial Group stated that

the contribution of the log was as important as the workflow automation as it enabled a Six-Sigma programme.

- It allows the two flexibilities that Heinl *et al* [53] 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 model at run-time by adding, removing or altering Business-Transaction Routing Sheets.

- Extended written use case language provided descriptions which can be understood by various stakeholders in a straightforward manner. Cox *et al* suggest that end-users do understand well the extended written use case language [26].

- Our approach enables pilots who were unfamiliar with the underlying routing to make complex routing decisions by concentrating on observations rather than activities.

4.8 Conclusion

We have thus shown that a use case oriented workflow engine was implemented in industrial settings and that the approach was well accepted by management, workflow configuration officers and workflow participants alike.

4.9 Reflections

Both BT and myself learnt much about the management of workflow in a single business environment. The lessons are described in this chapter and the chapter 3, and I am grateful to BT's management for allowing this publication. At a personal level I have learnt a few things which probably differ from the organisation's party line.

BTeP's more than eight years in operation demonstrate that a use case oriented workflow engine can handle large scale processes in the commercial settings of a financial institute. Until now BTeP managed more than $1,300,000$ work items and logged more than $600,000,000$ action steps.

In the Australian Financial Industry, most fund managers (BT Financial Group included) are remunerated by a fixed percentage of funds under management. Due to reduction in funds under management caused by the decline of international markets in the turn of the century, and below average investment performance, BT Funds Management implemented BTeP at a time when BT was under unprecedented pressure to reduce costs, and this led to BT Management's restless quest for operational efficiency. The analysis of business activity monitor logs was essential for the continuous support for the system by BT's management over the very long duration of the project. In particular the identification of expensive manual processes and the ability to prove that some of them could automated was key to the management support for the project.

But, the measurement and reporting capabilities were not always utilised for the improvement of client satisfaction. In particular establishment and measurement of internal service level agreements (SLAs) enabled certain departments to reduce their service level. I have noticed workflow items that violated external SLAs, while satisfying internal SLAs. While dispatching rules were always optimised to cater for external SLAs, individual workflow participants performance measurements were based on sometimes conflicting internal SLAs. That was usually when processing of work items required handover to other departments such as the call centre or the department that fixed complex issues such as backdating. In these cases the transfers from the main department to the second and back were padded with internal SLAs on each direction, practicably ensuring that the external SLAs will be breached.

For example, a deposit instruction has an external SLA guaranteeing that it will be processed within two business days, and internal SLAs of one day, guaranteeing that each department had to process it within 24 hours. Say, a deposit instruction queued for a day in front of the application team that required some qualification, say marital status, the work item had to

be routed to the client contact team, who had a next business day SLA and then returned the deposit instruction to the original team that started a new next 24 hours SLA. That say, the deposit instruction was processed within three business days, violating the external SLA by a day, but complying with the internal SLAs none the less.

As habits of the business analysts and workflow participants were formed before BTeP was implemented, they used to think in terms of activities only, rather than observation - activity pairs. Hence observations were often similar to activities. That is to say that what the pilots observed as properties of workflow items were activities that should be done. This, in many ways defeated the purpose of the separation between an observation and the activities the observation spawns.

For example think of a situation when a required tax file number was missing, and the activity that should have rectified this was to contact the client by phone and ask for the tax file number. In that case, the observation should have been: *"Tax file number missing"* and the activity should have been: *"Ask for tax file number"*. In the configured workflow, the observation had been *"Ask for tax file number"*. This could definitely not be observed.

Having inspected the code of BTeP I suspect that it would have probably been cheaper and faster to build BTeP upon a generic platform such as, for example, Microsoft SharePoint, or even the .Net or the J2EE platforms. As described in section 4.5 on page 104, BTeP's development team used the workflow capabilities of the engine to a very little extent, building a full use case oriented workflow engine upon a very basic FileNet hub-and-spoke workflow. Consequently BTeP's development team wrote scores of function points using IIS, JavaScript and SQLServer. BTeP's development team did gain some benefits from the functionality of FileNet's workflow engine, and the expertise of FileNet's integration partner, especially short-term time-to-market; however, over the life of the project it would have been cheaper for BTeP's development team to replicate the small amount of FileNet functionality BTeP's development team did use, rather than pay FileNet for the workflow engine licence and maintenance.

Treatment of scarce resources, such as highly qualified administrators, may have been sub-optimal, as the dispatcher often presented them urgent,

yet menial, tasks which could have been catered for by less skilled administrators who were often more plentiful, while the hard tasks accumulated and eventually violated the SLAs, while lesser skilled administrators where idle.

In this chapter I studied a single-case in detail. In chapter 3 I have specified what is a use case oriented workflow engine. A few questions remain open regarding the generic nature of a use case oriented workflow engine, namely: can the extended written use case language express complex workflows? and, can workflow participants and workflow configuration officers understand the extended written use case language, enough so, that they both can dynamically configure the workflow engine correctly? The following two chapters are devoted to the answering of these questions.

Chapter 5

Expressive Power

> Beware the Jabberwock, my son! The jaws that bite, the claws that catch! Beware the Jubjub bird, and shun The frumious Bandersnatch!
>
> Lewis Carroll [23]

Previously, we coined the term use case oriented workflow engine and described in detail a single implementation of a use case oriented workflow engine — BTeP. The experience of configuring workflow from use cases gained in an enterprise setting with BTeP, lead us to conjecture that the extended written use case language is very rich and expressive; enough so, that it could be taken as a general language for workflow modelling. In this chapter we test this conjecture by investigating the expressiveness of the extended written use case language.

5.1 Methodology

We employ two methodologies in this chapter. First we demonstrate that the extended written use case language can express patterns drawn from a rich workflow patterns library. Along the way, by means of examples,

we express some 15 workflow patterns using the extended written use case languages, of them three are pathological patters. We refer the reader to an appendix B where we articulated all 43 workflow patterns from a standard catalogue expressing them in extended written use case language. A strength of the pattern approach is that it provides a demonstration that the extended written use case language can express a variety of popular workflow scenarios and it can thus be applied by the industry. On the other hand, a weakness of this collection of case studies, is that is yet anecdotal in nature, and does not cover all possible workflow patterns.

The second methodology we employ in this chapter is to show that extended written use cases are able to express the union of every trace set of examples from another well-accepted language — the sound workflow net and the Petri net algebra which we summarised in section 2.1.1. We show that we can trace every sound workflow net by a written use case. The strength of this approach is that it is generic and shows that expressiveness of the extended written use case language is at least as great as another language. A weakness of this approach is that trace equivalence is a weak form of equivalence.

Thus we arrive at an original and general statement regarding the expressive power of the extended written use case language. Following this generalised statement, regarding the expressive power of the extended written use case language, we use Chapter 6 — Readability, to look at another aspect of power — the readability of artifacts employing the extended written use case language.

5.2 Patterns Approach

In this section we ask whether the extended written use case language's expressive power is sufficient, so that one can express, with the help of the language, common workflow patterns. We adopt a methodology previously deployed by Russell *et al* [97] and by Wohed *et al* [115], when they evaluated the richness of other workflow modelling languages – UML2.0 Activity Diagrams and BPMN. The Russell methodology is to see how a workflow modelling language would model each of the 43 workflow patterns which

were previously identified by the Workflow Patterns Initiative [96]. We carry out this agenda, articulating each pattern as a use case in appendix B on page 207. This appendix should be read with the patterns catalogue [96] at hand.

The idea of a catalogue of patterns originated in architecture in the work of Alexander [16]. It became widely accepted in the software industry following the publication in 1995 of the "Gang-of-Four" book listing patterns of object-oriented design [44]. For our domain of business process modelling, the seminal work has been published by the Workflow Patterns initiative [96] which started in 1999 and 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 aim of this initiative is to provide a conceptual basis for process technology. In doing so, they established a common vocabulary and agenda often used when discussing workflow subjects.

The Workflow Patterns Initiative identified 43 patterns, that seem to appear often in describing business processes. The patterns are named WPC-01 to WPC-43. Each pattern is presented in the Petri net language, in a fairly abstracted format (with steps labelled A, B, C etc); sometimes they are also illustrated with an example where the action steps have meaningful names.

Because the workflow pattern library is extensive and richly descriptive, it provides a good testbed for new approaches to business process modelling. A systematic presentation of all the patterns is beyond the scope of this chapter, but it can be found in appendix B on page 207.

We have simulated all the patterns using Petri net simulators, ensuring 100% coverage. In this section we describe the results of our simulation. We present written use cases that would be traversed in the same manner that the Petri nets would be traversed with identical input. In section 5.3 we will describe the algorithm we followed and prove that is general.

In this chapter we use several of these patterns to illustrate our claims; for example, the left hand column in Figure 5.3 presents some of the workflow patterns (however, we have made small modifications, shown in gray shading, to convert the patterns into the standard workflow net subset of

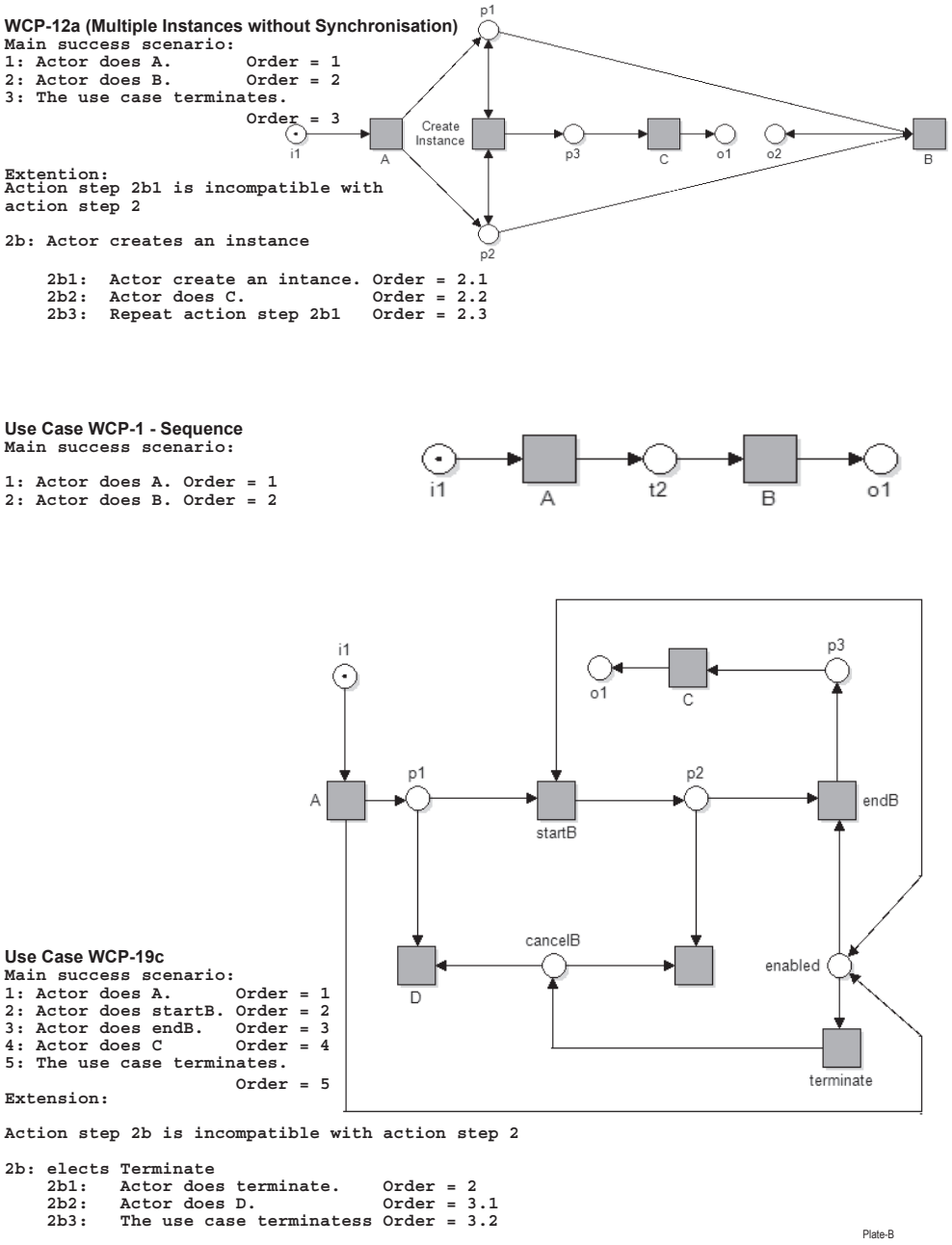
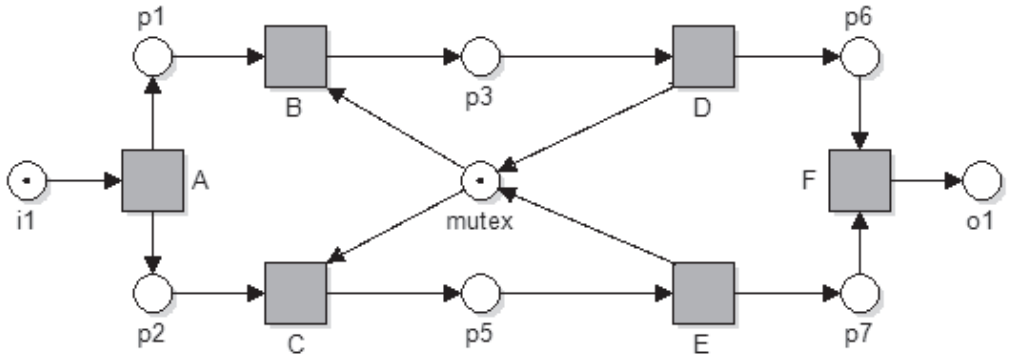


Figure 5.1 – WCP-01,WCP-12 and WCP-19

Petri nets, as described below). Other patterns are shown in Figures 5.1, 5.2 and 5.4.

In figures 5.1 and 5.2 on the next page we model five of the patterns in the extended written use case language.



Use Case WCP-39 (Critical section pattern)
Trigger: i1

Main success scenario:

```
1: Actor does A.    Order = 1
2: Actor does B.    Order = 2
3: Actor does D.    Order = 3
4: Actor does C.    Order = 4
5: Actor does E.    Order = 5
6: Actor does F.    Order = 6
```

Alternative Scenario

```
1: Actor does A.    Order = 1
2: Actor does c.    Order = 2
3: Actor does E.    Order = 3
4: Actor does B.    Order = 4
5: Actor does D.    Order = 5
                            6
```

Use Case WCP-40 (Interleaved Routing)
Trigger: i1
Main success scenario:

```
1: Actor does A.      Order = 1
2: Actor does B.      Order = 2
3: Actor does D.      Order = 2
4: Actor does C.      Order = 2
5: Actor does E.      Order = 2
6: Actor does F.      Order = 3
7: The process ends.  Order = 4
```

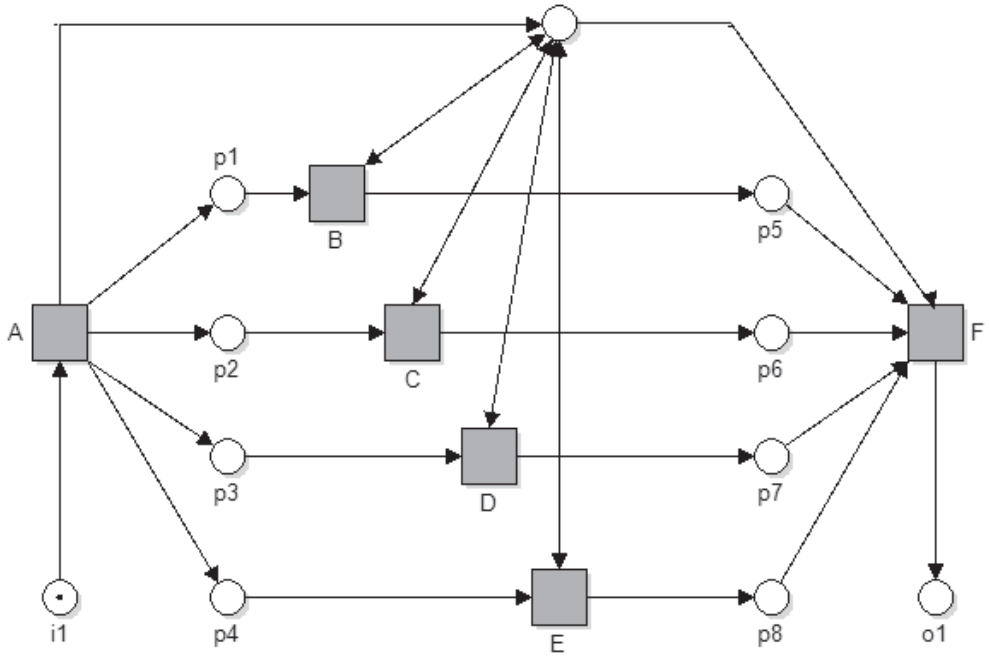


Figure 5.2 – WCP-39 AND WCP-40

5.3 Sound Workflow Nets Approach

Definition §5.1
induction, n. The bringing forward, adducing, or enumerating of a number of separate facts, particulars, etc., esp. for the purpose of proving a general statement [25, Accessed 23-November 2010].

Definition §5.2
deduction, n. The process of deducing or drawing a conclusion from a principle already known or assumed; spec. in Logic, inference by reasoning from generals to particulars; opposed to *induction* [25, Accessed 23-November 2010].

The inductive style of assessing expressive power which we have presented thus far is popular and often applied [115, 97]. However, one may want to go beyond a patterns catalogue and devise a somewhat more generic solution using deductive reasoning, as there is always the suspicion that there are workflow patterns outside the catalogue which are more demanding and that the extended written use case language would not be able to expresses them. To do so we call upon the well understood Petri nets mathematics and in particular upon the sound workflow nets. Recall that a workflow net is a sound workflow net *iff* for any case, the process terminates properly, i.e., termination is guaranteed, there are no dangling references, and deadlock and livelock are absent.

We show how one can mathematically obtain, from a sound workflow net, am extended written use case such that the union of the trace set of the sound workflow net is equal to the union of the trace set of the extended written use case.

When we argue about expressive power, we need to compare different models of workflow, in the same language or in different languages. Other research fields, such as concurrent programming, use many different notions of equivalence. For process modelling languages, researchers have considered several notions of equivalence [54].

Here we use trace equivalence, in which two models are considered equivalent if they generate exactly the same set of sequences of actions. This is

a weak concept of equivalence in general programming, but it is suitable for our purposes. In particular, trace equivalence is compositional – everything we can observe about the trace of an interaction of workflows can be deduced from the traces of the component flows. This was proved by Lynch [68, Thm 8.10] in an abstract model called I/O Automata, whose essential feature is that each action step is controlled in a single component of the system; more complicated equivalence definitions are needed only when the choice to do an action requires synchronisation, and so the "moment of choice" can be different in components with the same set of traces.

We note that workflow execution has the property that the workflow participant chooses an action from a worklist, and the system does not refuse any response returned by the workflow participant, so the result of Lynch holds for this model too for this reason, we claim that any trace equivalent description is sufficient to represent the essential features of a process modelling. Thus we prove that the extended written use case language can trace any reasonable business process, by showing how to trace a sound workflow net by a written use case set.

To this extent we take the opposite tack than that taken by Lee *et al* [66] when showing that every use case can be translated into a Petri net. That is, we show that sound workflow nets, can be traced by the extended written use case language.

Recall that sound workflow nets are a special class of Petri nets and that their set of possible markings is finite due to the workflow being a sound workflow net. Thus, we assume that we have been given a sound workflow net. We first show that it is possible to produce a trace equivalent use case set for this sound workflow net. The conversion we first show does not necessarily lead to an elegant or concise expression as use case set with our extension, just to some set of scenarios. We then show how to refactor the presentation we have first shown, into a more natural formulation. We conclude by providing an example.

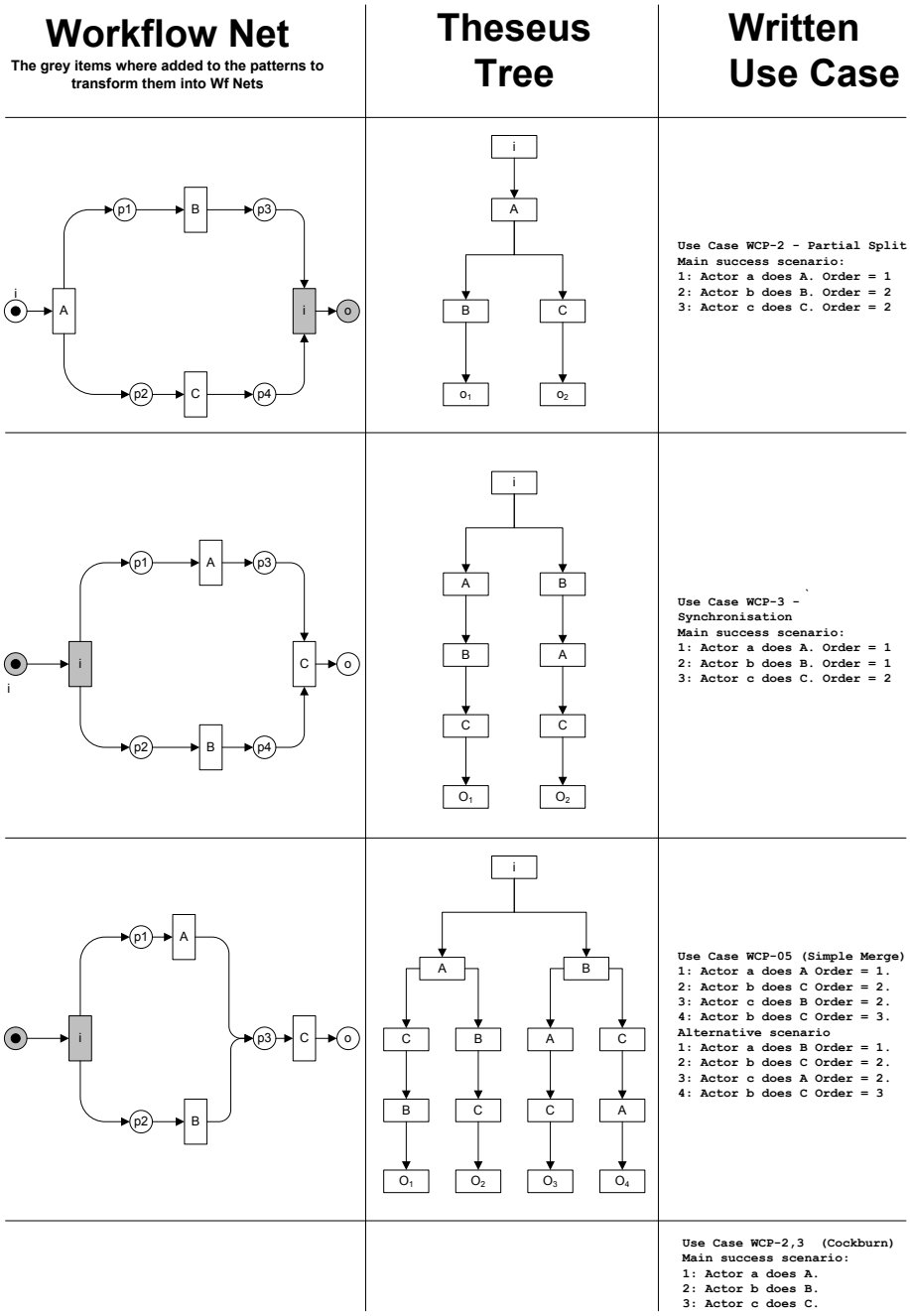


Figure 5.3 – WCP-2,WCP-3, and WCP-5

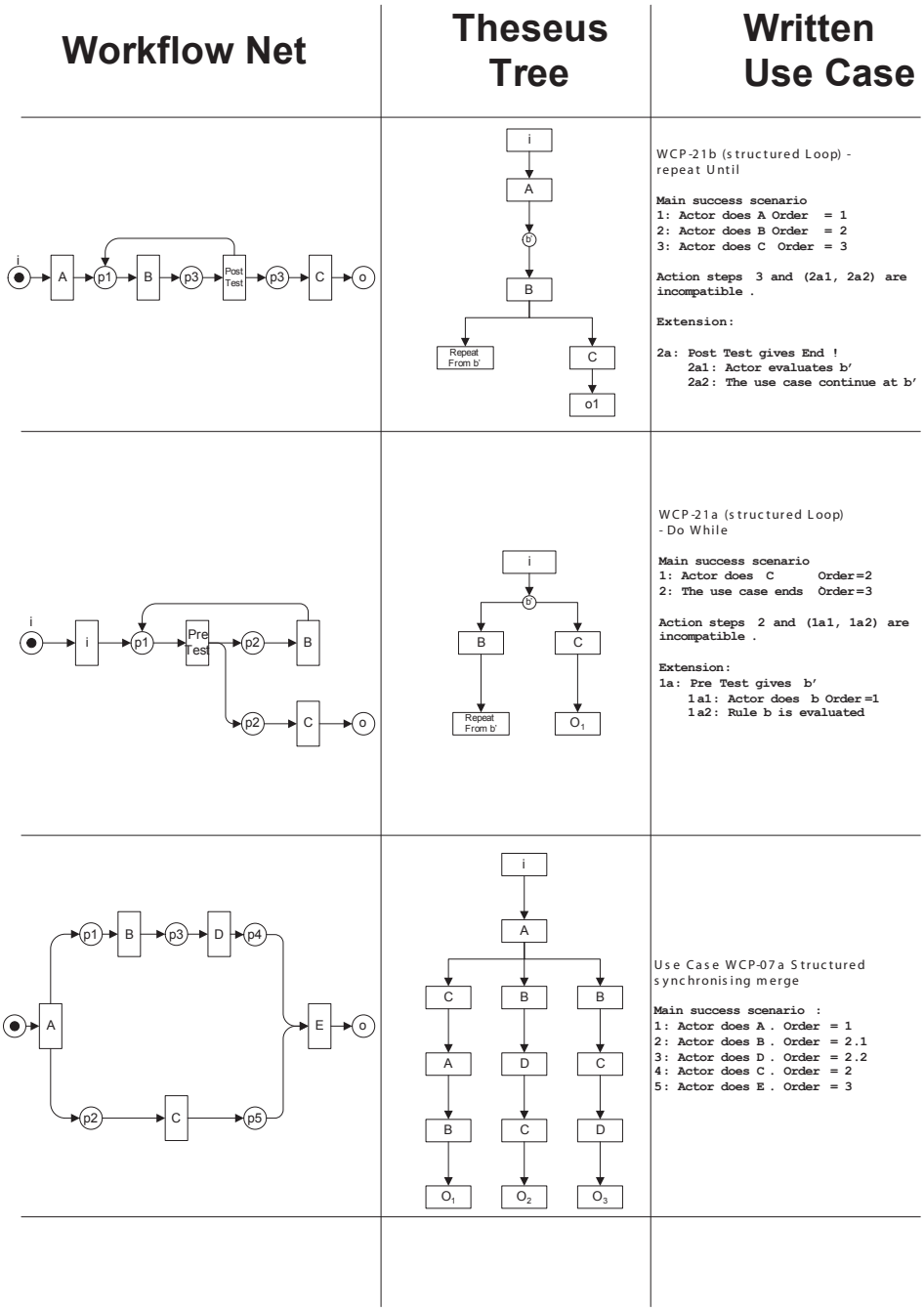


Figure 5.4 – WCP-21b,WCP-21a, and WCP-7a

Proof

Let us define *Theseus tree**, which captures the unwinding of the reachability graph of a net.

> **Definition §5.3**
> Theseus tree, n. A tree whose root is the start state of a graph S, the nodes are the set of reachable states from S, and the children of a node N are states that can follow N in one step and have not been included in the tree at the same or higher level.

The first phase of the conversion takes the sound workflow net, and produces a *Theseus tree*. Examples of as the unwinding of the reachability graph of a net into *Theseus trees* are provided in Figures 5.3 and 5.4 where we present six patterns drawn from the workflow patterns catalogue [96] and present each of them them as workflow net, Theseus tree and as extended written use case.

Lemma 1 – A business process that can be presented as a sound workflow net has a finite Theseus-tree

1. We consider every possible execution of the net; that is, we look at the set of sequences, each of which starts with the initial marking, and show how successive steps cause the net to reach a new marking. In each execution, we either reach the termination, or return to a marking that already occurred in the execution.

   This will definitely happen, because a Petri net (N, M_O) is said to be bounded if there is an upper bound on the number of tokens in any place for any marking reachable from the initial marking M_O. Thus, the total number of reachable markings (states) is finite [84].

   See figures 5.5, 5.6 and 5.7.

*We use the term Theseus tree because of the Greek myth describing how Theseus traced a trail through a labyrinth

2. When/if we return to a marking, we cut the sequence short at that point, with a "repeat from" indication that labels the earlier point where the same marking occurred.

3. When we reach the output marking of the net, we cut the sequence short at that point, with a "termination" indication.

4. This set of execution sequences can be built into a tree, a *Theseus tree*, by joining the sequences along a common prefix. Each path through the tree corresponds to an execution sequence; some of which end in termination leaves, and others end in repeat leaves.

We have now created the *Theseus tree* representation of the sound workflow net. This transformation will terminate, because the set of possible markings is finite due to the workflow being sound. □

Lemma 2 – Theseus-tree can be broken into a set of scenarios

Given the Theseus tree, we now define one scenario for each leaf of the tree, by concatenating the action steps that occur on the path from the root i, to that leaf.

1. Let us label the scenarios that lead to termination as $\overrightarrow{S}_{(1,\ldots,m)}$.

2. Similarly we label scenarios $\overrightarrow{R}_{(1,\ldots,m)}$ that lead to the repeat leafs. Each scenario is a sequence of transitions: action steps, with no extensions, yet.

3. Then we assign successive integers as order of processing identifier to action steps in scenarios $\overrightarrow{S}_{(1,\ldots,m)}$ and $\overrightarrow{R}_{(1,\ldots,m)}$ as described before.

4. We choose one scenario from $\overrightarrow{S}_{(1,\ldots,m)}$ and declare it as the main success scenario $\overrightarrow{S}_{(m)}$

5. If the *Theseus tree* splits at its root i, we may choose several other scenarios from $\overrightarrow{S}_{(k,\ldots,n)}$ and declare them as alternative scenarios $\overrightarrow{S}_{(k..n)}$.

From lemma 1, the *"Theseus tree"* being final, we will now present a solution for the main success scenario $\overrightarrow{S}_{(m)}$, and apply the same logic for the alternative scenarios $\overrightarrow{S}_{(k..n)}$. □

Theorem – Any sound business process can be traced by an extended use case set.

From lemma 2, we received a set of scenarios of which the main success scenario is $\overrightarrow{S}_{(m)}$. We will now refer to the remaining scenarios $\overrightarrow{S}_{(1,\ldots,l)}$ as alternate scenarios.

1. For each scenario in $\overrightarrow{R}_{(1,\ldots,m)}$, we start at the repeat leaf, $r_{(k)}$, traverse up to the repeat label and remove all the steps from after the label to the root i.

2. We declare the sets of transitions from the label and below as incompatible extensions to an arbitrary scenario that has its end leaf o, under the repeat label. I refer to the reason for the repeat label in the extension title[†].

We have now proved that sound workflow net can be translated into an extended Cockburn language. □

Refactoring

We now have a extended written use case language representation for the workflow net. To improve readability, we can repeatedly refactor the representation through either of the following changes:

1. From $\overrightarrow{S}_{(1,\ldots,l)}$, we collapse scenarios which differ only as permutations among a consecutive sub-sequence of action steps; and give all these action steps the same order-indication in the collapsed scenario.

2. To further the reduction , we start from an arbitrary output leaf, $o_{(k)}$, traverse up to the first junction above that output leaf, j, and remove all but one of the paths leading up from the junction j to the root i. I declare the remaining sets of transitions from the junction j down to the leafs $o_{(k,\ldots,m)}$ as incompatible extensions to the same scenario and refer to the reason for the junction j in the extension title.

[†]Failure to execute this operation will indicate the workflow has an infinite loop.

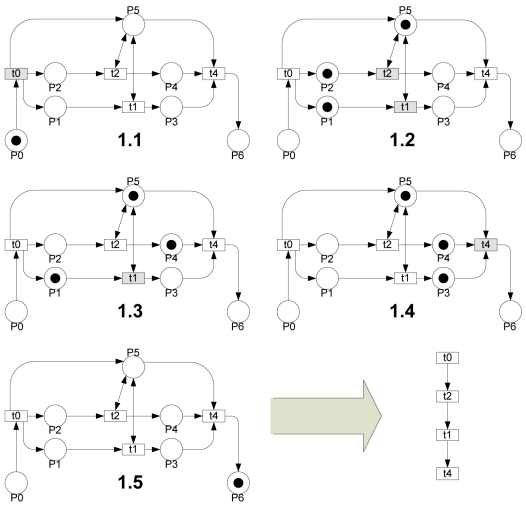


Figure 5.5 – Producing a Theseus tree from a Petri net — Step one of two — The move from $t0$, to $t2$, to $t1$, and to $t4$. Please note that the double headed arrow denotes two transitions, each on one direction.

3. We expect to repeat steps 1 and 2 while the readability of the overall use case increases.

Example

In figures 5.5, 5.6 on the next page and 5.7 on the following page we go through a simplified version of workflow Control-Flow Pattern WCP-40 [96, Page 73]. We first explore the only two possible paths through the net. We then merge the paths into a Theseus tree. We conclude by producing a use case set.

5.4 Unsafe Patterns

We have shown that the extended written use case language is expressive for a wide class of safe workflows. We now consider whether it can also model some pathological patterns. We find that it can.

In Figure 5.8 on page 130 we present use cases that are not safe, namely that have deadlocks, that have unreachable nodes or have infinite loops. In that figure use case #1 enters a dead lock when the data entry person

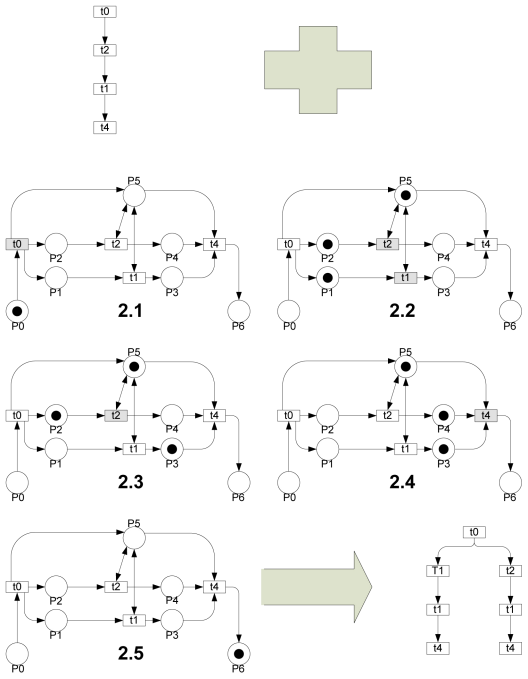


Figure 5.6 – Producing a Theseus tree from a Petri net — Step two of two — Starting at the already described $t0$ we branch to $t1$, move to $t2$, and end at $t4$. Please note that the double headed arrow denotes two transitions, each on one direction.

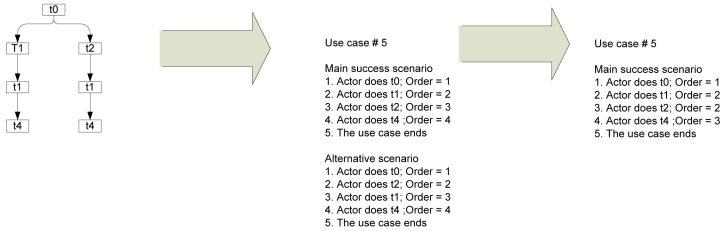


Figure 5.7 – Here we take the Theseus tree, translate it into the extended written use case language and optimise it.

finds that the indexing of the document does not match the data on the document, in use case #2 action step 1a1a1 inaccessible and use case #3 enters an infinite loop when the data entry person finds that the indexing of the document does not match the data on the document.

It is thus possible to our notation to create workflow patterns which are not safe, that is, have dead locks or infinite loops or unreachable nodes. This is a risk that can be mitigated. Detection of these undesirable behaviour can be done using tools constructed for Petri net. Recall that Lee *et al* have shown that use case can be transformed into Petri nets [66] and that the safety of Petri nets may be assessed using tools such as these described by van der Aalst [3] and implemented by Fahland *et al* [37].

5.5 Conclusion

Cockburn says that he does not care much about modes of parallelism and variants of OR statements (indeed he does not differentiate between OR and XOR); he has a bigger fish to fry - communication, and by extension readability [24, Page 26]. However, since we do care about parallelism, we have added the order structure, and since we do care about the difference between XOR and OR, we have added explicit syntax to carter for the differnce between XOR and OR.

By now we know that:

1. Sound workflow nets are safe [3].
2. Petri nets can be tested to see if they are sound workflow nets [3].
3. Every use case can be expessed as a Petri net [66].
4. Every reasonable workflow can be expressed using our proposed extended written use case language as a use case (from our proof above).
5. Every reasonable workflow can be expessed with the help of our proposed language as a use case. Indeed, that use case can be expessed as a Petri net. That Petri net can, in turn, be tested for to see if that Petri net is a sound workflow nets. If that Petri net is indeed a sound

```
Use-Case 1 - Process a retirement application
Primary actor - Data entry person who keys in the application.
Main success scenario:
1. The WF system presents a work item document to the data entry person.
2. The Data entry person keys the details from the work item document to
   the processing system.
3. The data entry person releases the document to the WF system.
4. The process ends

Extensions:
    1a. The data entry person finds that the indexing of the document does not
        match the data on the document.
        1a1. Expecting an indexing swap, the data entry person retrieves a
             document indexed like the details on the document.
        1a2. The data entry person changes the indexing of the first document.
        1a3. The data entry person changes the indexing of the second document.
        1a4. The data entry person releases the first document.
        1a5. continue step 2 with the new document.
```

```
Use-Case 2 - Process a retirement application
Primary actor - Data entry person who keys in the application.
Main success scenario:
1. The WF system presents a work item document to the data entry person.
2. The Data entry person keys the details
3. The process ends

Extensions:
    1a. The data entry person finds that the indexing of the document does not
        match the data on the document.
        1a1. The data entry person fixes the issue.
        1a2. The use case continues at action step 2

             1a1a. The  entry person finds that the indexing of the document does
                   match the data on the document
                   1a1a1. The use case continues at action step 3.
```

```
Use-Case 3 - Process a retirement application
Primary actor - Data entry person who keys in the application.
Main success scenario:
1. The WF system presents a work item document to the data entry person.
2. The Data entry person keys the details
3. The process ends

Extensions:
    2a. The data entry person finds that the indexing of the document does not
        match the data on the document.
        2a1.  The use case continues at action step 1.
```

Figure 5.8 – Unsafe patterns — Here we present use cases that are not safe, namely that have deadlocks, that have unreachable nodes or have infinite loops. Use case #1 enters a dead lock when the data entry person finds that the indexing of the document does not match the data on the document, in use case #2 action step $1a1a1$ inaccessible and use case #3 enters an infinite loop when the data entry person finds that the indexing of the document does not match the data on the document

workflow nets, that Petri net is safe, and by association the use case is also safe (from 1,2,3 and 4).

6. Every reasonable workflow can be expessed with the help of our proposed extended written use case language and be tested for safety (In summary). □

We derive the sense of what is 'a reasonable workflow' and what is not from the workflow coalition's own patterns library [96] and from our experience in BT Financial Group, where we collected a rather large sample of work items over eight years. Indeed our sense is that most industrial work items trace short, yet highly variable paths. As for safety, we focus on proving that our proposed extended written use case language can express every possible safe workflow. We do not prove that our proposed language can express every possible non-safe workflow. Indeed, for obvious reasons, we do not care if we cannot express some possible non-safe workflows. However, by expressing workflows using our proposed extended written use case language, translating them into Petri nets and applying established Petri net verification tools, we can detect if the workflows that we do express are safe or not.

We have thus shown that the extended written use case language is quite expressive, at least as much so as the incumbent graphical language.

5.6 Reflections

The work I conducted at BT-Financial Group exposed me to more than 350 use cases. While these use cases were drawn from one domain – financial services, and were not at all a representative sample for workflow patterns, they led me to conjecture that the extended written use case language is very expressive. To demonstrate this, I first conducted a patterns based evaluation which show that the language can express a variety of patterns taken from a multitude of industries. In the process I have developed a conversion algorithm, translating patterns from sound workflow nets into use cases. Proving that the algorithm can be applied so that the extended written use case language can express *every* reasonable workflow pattern

gives further confidence in my conjecture. I then explored the outer range of the notation's expressive power and found that it can express unsafe patterns.

My interest in Petri nets stems from the research community extensive experience with of Petri nets to formally model workflow in the organisation and between organisations [52]. There are many variants of Petri nets. In this dissertation, I focus on one class of Petri nets — the sound workflow nets, which are especially appropriate for modelling of workflow [6, 32]. An important attribute of Petri nets, on which I rely in section 5.3, is the reachability graph of a net.

I decided to compare the extended use case language to sound workflow nets, rather than to *de facto* standards such as BPMN or UML Activity Diagrams which are more widely accepted. I made this choice not because sound workflow nets are modern, readable, maintainable, or accepted by the industry as a user interface, but because, as van der Aalst *et at* have shown, sound workflow nets are known to be expressive for workflow modelling [4].

One issue still requires investigation, though. I did not assess the usability of the extended written use case language. The following chapter is devoted to assessing the effectiveness of the language in conveying the writer's intent correctly.

Chapter 6

Readability

> 'It seems very pretty,' she said when she had finished it, 'but it's RATHER hard to understand!'
>
> Lewis Carroll [23]

In this dissertation we propose to feed models expressed in the written use case language into workflow engines as a workflow modelling language. We coin the term use case oriented workflow engine in Chapter 3, study an industrial implementation of a use case oriented workflow engine in Chapter 4, and assess the expressive power of the extended written use case language in Chapter 5. One fundamental question still begs answering:

> *Are artifacts written in the extended written use case language more readable by a wide cohort of readers than the incumbent graphical languages?*

6.1 Methodology

We assess the readability of the extended written use case language by comparing it with that of BPMN, a language we use as a proxy to a class of

incumbent workflow modelling languages. We measure the readability of these languages by two groups of participants who serve as proxies for different communities: for subject-matter-experts and workflow participant on one hand; and people trained in business process management such as business analysts or workflow configuration officers on the other hand. While doing so we assess several predictors for the graphical or sentential success of readers.

The methodology we deploy at this chapter follows the suggestions of Wohlin *et al* [116]. An advantage of this methodology is that it is systematic and tailored for experimentation in computer science settings. The presentation format that Wohlin *et al* recommended, and that we adopted, necessitates repetition of few definitions from earlier chapters of this dissertation.

In section 6.3 — Problem Statement, we explain the particular question that we address in this chapter, giving the established theory that relates to readability of languages. We follow by a short revisit of the Business Process Management domain and the languages we compare. In section 6.4 — Experiment Planning, we discuss the appropriate research methods for our evaluation, thus we position our readability research within the wider research framework. We also list the hypotheses that our experiment tests. Then, in section 6.5 — Experiment Operation, we describe our experiment, present the participants, the instruments that the participants worked through, and the procedure the participants followed. We also describe how we controlled the experiments to extend the reliability of the results. Finally in section 6.6 — Data Analysis we present our experimental results and the statistical tests we applied.

6.2 Background

Readability by diverse groups of people is crucial for a workflow model to effectively bridge the various communities involved in the round trip of development, enactment, analysis and improvement of workflow. Errors introduced early in the development process are commonly the most expensive to correct, and a review of workflow models by a wide audience

would detect these errors, should the artifacts be readable. Let us consider a situation where business representatives approve a workflow model, and at a later time approve an implementation of the workflow. A prudent risk manager may be concerned as to whether the approvals are made on the basis of correct communication between the different stakeholders, including end users, business analysts, modellers and workflow configuration officers. If the models are not correctly understood by all stakeholders, the project's outcome is threatened. Mitigation of communication risk gives rise to such practices as peer review, or prototyping, or even doing away with specifications. Here we address the core of the issue – readability.

6.3 Problem Statement

In this chapter we present an experimental assessment of the readability of the extended written use case language by comparing its readability with that of BPMN. Each language has been proposed for modelling in the domain of business process management. BPMN is graphical, with the process structure captured in labelled diagrams; the other, our proposed extended written use case language, is textual. The extended written use case language applies stylised natural language text.

Previous work offers insights into the readability of such languages, and provides the theory underpinning the particular questions we designed the experiment to answer. Many of the languages used to represent workflow models are graphical, laying out shapes and connecting lines (with attached labels) on a page in order to convey information about software structure or behaviour. One of the early graphical languages in software engineering was the flow-chart, and more recently the diverse diagrams that make up UML have been widely accepted. It is commonly held that diagrams and visual languages are easy for humans to understand; a seminal argument for the benefits of visual over textual language was given by Larkin and Simon [64], who reasoned that text is limited to a linear order, whereas a diagram allows more information to be carried by the spatial arrangement of different elements in the language. In contrast to Larkin and Simon's arguments, Moher *et al* [76] looked at several ways to express program

structures in text and in diagrams (Petri Nets); they found "for our tasks, graphics were no better than text , and in several cases were considerably worse".

Indeed, programming languages, are usually textual, and graphical programming has not found much uptake. Gruhn *et al* found that graphical models often contain mistakes that are avoided in textual programming, such as using OR-join in place of XOR-join or AND-join [49]. In this chapter we offer an empirical check of these conflicting views of the readability of graphical and textual languages, in a particular domain and with particular languages of each sort. By checking these views we arrive at a conclusion regarding the readability of the extended written use case language, a key building block of a use case oriented workflow engine. We carefully evaluate whether each language does convey useful information to readers, and compare the extent of information gain by readers from the two languages. We thus test hypotheses which claim that the extended written use case language is effective, that the graphical language is effective, that the extended written use case language is more effective than BPMN, and that BPMN is more effective than the extended written use case language.

As noted, we see it as vital that readability should be evaluated for the different stakeholder communities among whom communication is to take place. Thus the experiment was with participants divided into groups. Some participants represent IT professionals such as business analysts or workflow configuration officers, who have previous training in workflow models, and others are representative of a broader community in the organisation. We regard a hypothesis as valid only if we see significant support for it among each kind of participant considered separately.

Green [47] observed that while different languages can achieve identical ends, where the information structures they use is different, they facilitate different cognitive processes. In our case, while the languages can present the same information as showed in Chapter 4 , the extended written use case language may be better at convening a multitude of exceptions and BPMN may be better at convening a multitude of nested loops. As each language highlights some types of information while obscuring other types, each language may facilitate some tasks while making others harder. There-

fore, the languages may not be absolutely good, but good only in relation to certain tasks. If so, we expect that presenting the same information using two languages would increase comprehension.

We are also inspired by analogy with other situations, such as education, where text and diagrams are often used together, to reinforce a message through different media. Thus the research we present in this chapter also studies the effect of giving readers models of a workflow in both languages, one after the other. We test the hypotheses that doing so conveys greater information than presenting the workflow model in only one language.

Finally, we are interested in whether there are personal features about different people, which could predict whether they would receive information well though one language or the other. Vessey [110] differentiates between a problem representation using a language and its mental representation. Reading is thus a transformation from one representation to another. A good fit of a language to a problem-solving style, would simplify the reading process by requiring less transformation. Ideally there should be no transformation from the language based representation to the mental representation. Hence, a good cognitive fit of a language to a thinking style would lead to an effective and efficient problem-solving process. We would thus expect to see difference between the understandings of the two languages depending on the thinking styles of the readers. In our case, since we are comparing a graphical language with a sentential one, we explore whether it mattered if a reader had a preference for sentential or graphical information in other contexts; we also consider experience with a similar language, as a possible predictor of the effectiveness of a language.

The rest of the chapter shows in detail the experiment we performed, to compare the readability of extended written use case language and BPMN, for conveying information about workflow to readers from different communities.

6.4 Experiment Planning

We use an experimental approach to investigate the problem identified in Section 6.3. Having described in section 2.1.1 on page 21 a taxonomy of

research methods that can be applied, we will now position our research within this taxonomy.

In this chapter we present a between-grammar study, that measures understanding of a domain. For the within-grammar comparison we vary the training background of participants. We vary the treatment which we randomly, and double blindly, assign to participants. We use both between subjects and within subjects comparisons. The multiple subjects are students, the multiple objects are taken from toy problems, and the experiment occurs offline. Details of the experiment procedure are in Section 6.5.

6.4.1 Hypotheses

We present the explicit hypotheses that are tested in our experiment. First we determine whether each language can be read effectively.

H_1**:** Information about workflow is conveyed to business analysts and subject matter experts from reading a model in the extended written use case language.

H_2**:** Information about workflow is conveyed to business analysts and subject matter experts from reading a model in BPMN.

We next offer two (mutually contradictory) claims about the comparison between the languages. These go to the heart of the debate about suitable choice of language.

H_3**:** More information about workflow is conveyed to business analysts and subject matter experts from reading a model in the extended written use case language than from reading a model in BPMN.

H_4**:** More information about workflow is conveyed to business analysts and subject matter experts from reading a model in BPMN than from reading a model in the extended written use case language.

Two further hypotheses concern the value of providing an additional presentation of the same information in a different style of language.

H_5: More information about workflow is conveyed to business analysts and subject matter experts from reading a model in the extended written use case language followed by a corresponding model in BPMN, than from only reading a model in the extended written use case language.

H_6: More information about workflow is conveyed to business analysts and subject matter experts from reading a model in BPMN language followed by a corresponding model in the extended written use case language, than from only reading a model in BPMN.

As mentioned explicitly in each hypothesis, we are interested both in the community of business analysts, and in the community of subject matter experts from the broader business context. Our experiments used two groups of participants as proxies to these communities, and we consider a hypothesis validated only if both groups show the effect. One type of participants can be considered as proxies for business analysts (BAs), since they have received explicit training in business process modelling and flowchart notations. Post-graduate students (like these) have been previously found to be adequate proxies for analysts with low to medium expertise levels [46, 91, 94].

6.5 Experiment Operation

The common method for assessing readability, in a research concerning human computer interaction, or indeed in medicine or law, is to give people a document to read, and measure how much they know by seeing how well they answer questions whose answers come from the document. The result is the sum of three components: initial domain knowledge, contribution from the document, and chance. We refer to this total measurement as absolute readability ($QSet_n$).

Definition §6.1
absolute readability ($QSet_n$), n. The measured knowledge of a participant after reading an artifact.

We measure readability by concentrating on the average contribution from the document, which is the component that the workflow modeller controls. We refer to this measurement as *relative readability*. Our experimental procedure aims at reducing chance, gauging participants' initial domain knowledge, and gauging the absolute readability ($QSet_n$). We arrive at the relative readability of the workflow model by subtracting the initial domain knowledge from the absolute readability ($QSet_n$).

6.5.1 Participants

196 participants, all post-graduate students, were drawn from three universities. 129 participants were industrial engineering students from Eindhoven University of Technology, The Netherlands (TU/e). 26 participants were advanced business process management and enterprise systems students from Humboldt-Universität zu Berlin, Germany (HU). They were encouraged to take part as the experiment was relevant to their studies. Both types of participants can be considered as proxies for business analysts (BAs), since they have received explicit training in business process modeling and flowchart notations. Post-graduate students (like these) have been previously found to be adequate proxies for analysts with low to medium expertise levels [46, 91, 94].

The remaining 41 participants were students following various courses in the University of Sydney, Australia (USYD). They were recruited by advertisements on noticeboards in cafeterias, and paid *AU*$20.00 for their effort. Participants from USYD are considered as proxies for business users (BUs) without training in flow-charting. They come from a broad range of disciplines, and are likely to act in that role a few years after joining the workforce. Note that the preliminary knowledge of business user participants is not relevant, as will become clear from the discussion of the controls that have been applied in this experiment.

6.5.2 Instruments

We asked each participant to follow through a workbook, a variant of it is available in Appendix A on page 167. A supplementary technical report

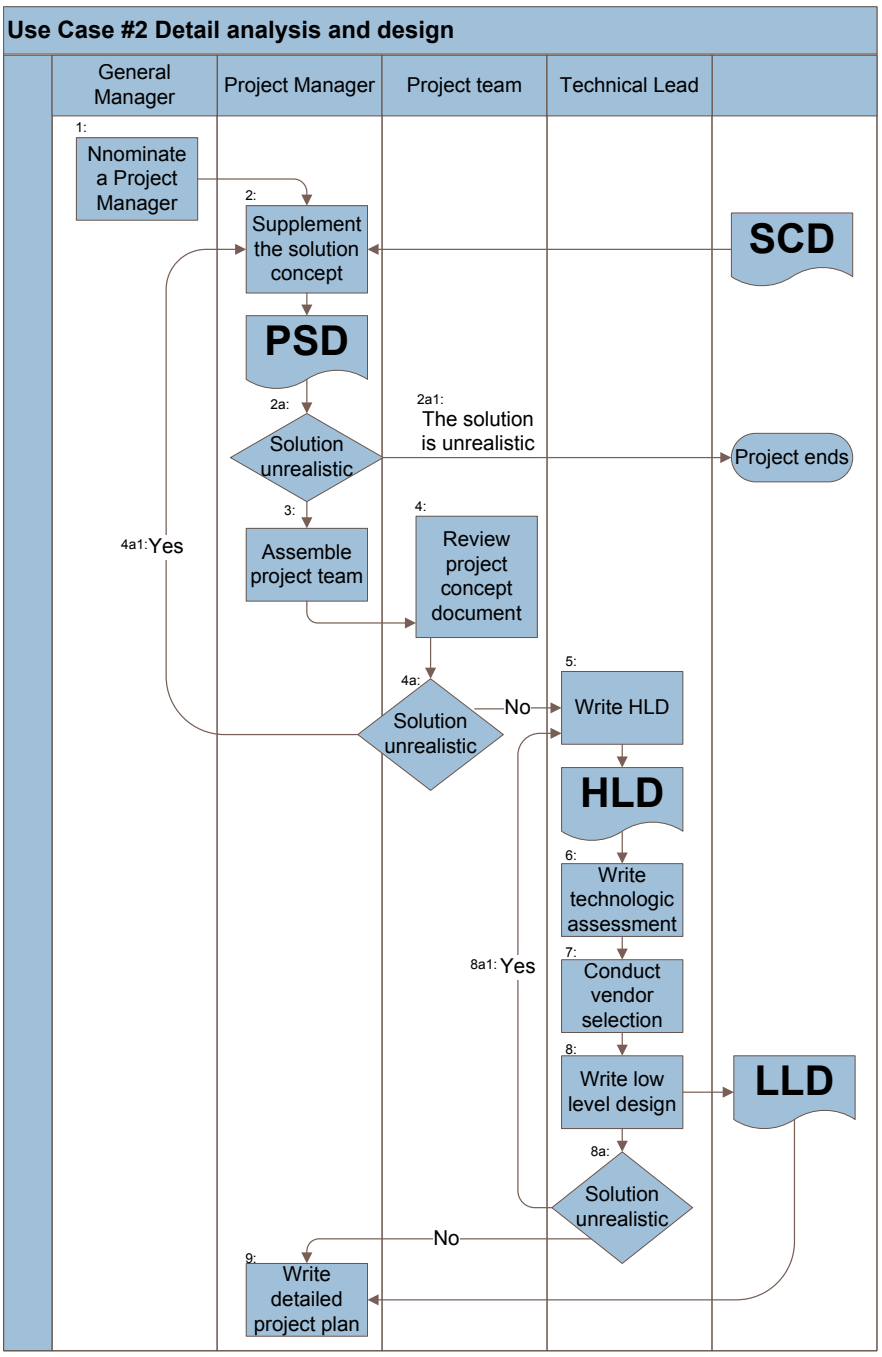


Figure 6.1 – BPMN **Example.** The participants' hand book, a variant of it is available in Appendix A on page 167, had six BPMN diagrams of equal complexity.

```
Use Case #2 Detail analysis and design
Use case Scope: High level
Trigger: Solution Concept Document (SCD) is completed and approved
Primary Actor: Project Manager who has to assess the project cost in detail
Actors: Project team, Technical Lead

Main success scenario
1:  General Manager nominates a Project Manager.                    Order = 1
2:  Project Manager supplements the solution concept document
    with detail, writing a Project Scope Document (PSD).            Order = 2
3:  Project Manager assembles the project team, including direct
    reports, customers, suppliers and auditors.                     Order = 3
4:  Project team reviews Project Concept Document.                  Order = 4
5:  Technical Lead writes High Level Design (HLD) evaluating
    several design avenues.                                         Order = 5
6:  Technical Lead writes an assessment of technology and methods
    selecting one of the design alternatives.                       Order = 6
7:  Technical lead conducts vendor selection.                       Order = 7
8:  Technical lead writes a Low Level Design (LLD) (Solution
    Design, Support impact, Risk assessment).                       Order = 8
9:  Project manager writes Detailed Project Plan using input from
    the LLD (Time Line and Resources in Microsoft Project, Cash flow,
    Risk Management Plan).                                          Order = 9

Extensions:
    2a: The project manager finds the Solution Concept Document unrealistic
        2a1:    The project ends
    4a: The project team finds errors in the Project Concept Document
        4a1:    The use case starts at action step 2
    8a: While writing the LLD the technical lead finds errors in the HLD.
        8a1:    The technical use case continues at action step 5.
```

Figure 6.2 – extended written use case language **Example.** The participants' hand book, a variant of it is available in Appendix A on page 167, had six extended written use case language diagrams of equal complexity.

includes a sample instrument we used in our experiment [88]. The preamble to the workbooks was a disclosure statement. It was followed by a privacy statement. At the core of the workbooks resides a description of a certain business issue. The workbook also includes a placebo, a solution to the business issue, three identical questionnaires, a preferences survey, and a demographic survey. The length of the workbook is 39 printed $A4$ pages.

Being descriptions of a toy problem, the workflow models were shorter, poorer in red herrings, less ambiguous, richer in workflow patterns, yet more consistent than real life workflow models. The workbook articulated the genuine workflow solution to the business issue twice, once employing eight $A4$ size BPMN diagrams similar in complexity to that of the diagram in Figure 6.1, and once employing use cases of similar complexity to that of the text in Figure 6.2. Each questionnaire had six multiple choice questions about the genuine workflow solution to the business issue.

6.5.3 Procedure

In the workbook, we asked the participants to complete the following procedure:

1. Read a disclosure statement explaining the experiment's goals, the tasks, and the participants' privacy.

2. Read, and optionally, sign a consent form. The participant could opt out at any stage.

3. Read a description of a business issue that a project office in a hypothetical financial services company faces.

4. Read the placebo that describes the financial services that the hypothetical company offers. It includes no information related to the business issue that the project office faces.

5. Answer a multiple choice questionnaire asking about factual matters concerning workflow models that solve the business issue the project office faces, a solution that was not presented yet to the participants.

6. Read an artifact that models the workflow in one language.

7. Answer the same questionnaire for the second time.

8. Read a second model, presenting the same workflow in the other language.

9. Answer the same questionnaire for the third time.

10. Fill in a preferences survey.

11. Fill in a demographic survey.

In order to compare the impact of different languages, we used two workbook types; in one type we used one language in Step-6 and the other language in Step-8; the other workbook type reversed the order in which the languages were given. We refer to one workbook type (and to the condition of participants who receive this workbook) as "BPMN first"; the other condition is "written use cases first".

6.5.4 Pilot Study

As per the recommendation of Dix *et al*, we conducted a pilot study prior to the large scale distribution of workbooks to participants [34, page 350]. The study involved two groups, of seven members each, a sufficiently large number according to Nielsen [82]. One group has been drawn from the University of Sydney, Australia, School of Information Technologies. The second came from practising members of the information technology industry. We issued the workbooks to the participants and observed their behaviour. Of note is an observation that, being experienced students, some participants read all the instruments before answering questions. Later we discouraged this behaviour by adding a specific guideline to the disclosure statement.

The pilot study helped us fine-tune the instructions in the workbooks. It confirmed that the workbooks were complex, but not too complex, and that participants can complete the tasks we listed in the workbooks in the allotted 50 minutes. We did not include the 14 returns from the pilot study in our statistics.

6.5.5 Control

To arrive at reliable results, from which we will be able to generalise beyond the narrow scope of the experiment, we instituted several controls.

Initial Domain Knowledge

In Step-5 of the experiment, to convince the participants to answer questions after only seeing the placebo, we stated that:

> *"The philosophy of this design language is that one must understand the products of a company to understand its processes"* [88, Page 113].

We thus used the questionnaire presented in Step-5 to measure initial domain knowledge. To neutralise initial domain knowledge, we subtracted the placebo score from both the written use cases and the BPMN scores, and arrived at the average contributions of the two languages. In Step-7 of the experiment we measure the absolute readability ($QSet_n$) of the first language, and in Step-9 we measure the absolute readability ($QSet_n$) of a presentation through both languages in sequence. Later we refer to the results of the questionnaire presented in Step-5 as *Placebo*, to the results of the questionnaire presented at Step-7 as $QSet_1$ and to the results of the questionnaire presented in Step-9 as $QSet_2$. We now define:

Definition §6.2
primary contribution ($Cont_1$), n. $Cont_1 \equiv QSet_1 - Placebo$.

Definition §6.3
secondary contribution ($Cont_2$), n. $Cont_2 \equiv QSet_2 - QSet_1$.

Realistic Patterns

We could have equalised the initial domain knowledge by using nonsense words, as Lewis Carrol did in Jabberwocky [23], or by using a schematic process as other experimenters did [72]. However, as we wanted to mimic reality as closely as possible, we used a meaningful real life business issue

because humans process meaningful information in a manner different to nonsense or schematic information [99, page 133].

Information Equivalence

The usage of real life business issues required us to ensure equivalence between the two models.

> **Definition §6.4**
> information equivalence, n. Two representations are informationally equivalent if all of the information in the one is also inferable from the other, and *vice versa* each could be constructed from the information in the other [64].

When presenting two different workflow models to participants, whether they are within-grammar or between-grammar, the workflow models should include equivalent information. Larkin *et al* also speak about computational equivalent representations, but the criteria to evaluate that equivalence are subjective [45, 102].

For example, consider the navigation from point A to point B we illustrate in Figure 6.3. Verbally we may phrase a statement regarding the navigation as: *"walk south east* $400m$*, turn left and continue* $20m$*."* Graphically, compromising the information equivalence, we may add information such as the names of streets passed and surrounding streets. We would have achieved information equivalence had we only given the L shaped curve as the graphical instruction.

To ensure the information equivalence of the BPMN and the extended written use case language workflow models, we reconciled the models with each other, ensuring that the use case sets and the BPMN diagrams were (i) logically identical, (ii) included the same information, and (iii) included the same amount of information. Note, that using Moody's terminology, the written use case language suffers from symbol deficit [77].

Allocation Bias

> **Definition §6.5**
> allocation bias,n. An undesirable property of an experiment

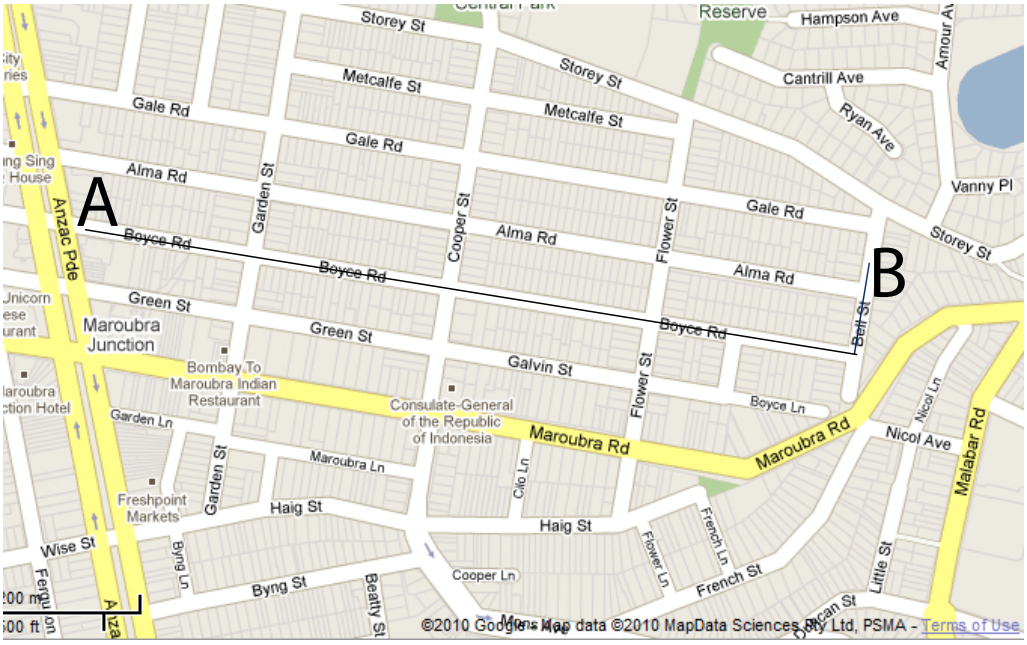


Figure 6.3 – Information Equivalence

> where by the officer executing the experiment effects the results of the experiment by assigning participants to one of the experimental groups on the basis of familiarity with both the participants and the experimental groups.

To prevent allocation bias, we applied double blind sampling. We randomly sorted the workbooks and placed these in sealed unmarked envelopes, thus preventing unintentional bias in the allocation of participants to groups according to perceived verbal or graphical aptitude. A side effect of this procedure was that we could not ensure that the size of the two groups was identical.

Objective Result Processing

To ensure the correct interpretation of participants' answers, the questionnaires concentrated on the knowledge domain, the lowest level within Bloom's cognitive section [62]. We asked questions such as: *"What happens if stakeholders change the project"* or *"What condition determines when a build is reiterated?"*.

Anonymity

To ensure anonymity, participants were asked not to write their names on the workbooks and to remove the disclosure statement and the signed privacy statement from workbooks before commencing the experiment. To further extend the perceived anonymity of the participants, we added *"I do not wish to answer this question"* options to each question in the preferences and the demographic surveys.

Reflection of Participants' Views

To allow the participants to fully articulate their understanding of the workflow solution, or even criticise it, we included the following statements *"I do not know"*, *"The workbook does not supply information needed to answer the question"* and *"None of the above"* option to each question in the questionnaire.

Chance

To reduce chance we randomised the choice sequence among the concrete answers. As well we offered seven options for every question, rather than the conventional four.

Sufficient Time

The 50 minute time constraint only allowed six questions in each questionnaire.

Read Forward Prevention

To ensure that participants answered questionnaires one by one, immediately after reading the appropriate instrument, we asked participants not to read ahead. We decided against issuing mini workbooks to participants one by one, as the synchronisation of this threefold increase in anonymous workbooks would have risked the integrity of the experiment.

6.6 Data Analysis

Our main, randomly controlled, *independent variable* was the order of the treatments namely: (*i*) BPMN first and written use cases second, and (*ii*) written use cases first and BPMN second. Another aspect that varied was the group to which each subject belonged (proxies for BAs, or proxies for SMEs); however we regard this not as an independent variable within one experiment, but rather, we consider a hypothesis on each group of subjects, and ask for its validity on both groups.

Our *dependent variables* were the *primary contribution* ($Cont_1$) and *secondary contribution* ($Cont_2$) to the readability. Recall that primary contribution ($Cont_1$) is the score on the second questionnaire *minus* the first questionnaire (the placebo). Similarly, secondary contribution ($Cont_2$) is the change between the second and third questionnaire (due to seeing the second language). We use *R* and *Stata* [93, 104] for statistical analyses.

Table 6.1 – Results From the Three Universities

University	First artifact	Score	$\overline{x}$	s	n
TU/e	BPMN	*Placebo*	1.96	1.09	74
		$QSet_1$	3.41	1.32	
		$QSet_2$	3.23	1.37	
	UC	*Placebo*	1.78	1.10	55
		$QSet_1$	2.73	1.38	
		$QSet_2$	3.16	1.36	
USYD	BPMN	*Placebo*	2.05	1.22	19
		$QSet_1$	2.47	1.39	
		$QSet_2$	2.53	1.07	
	UC	*Placebo*	1.86	0.77	22
		$QSet_1$	2.68	1.09	
		$QSet_2$	3.55	0.86	
HU	BPMN	*Placebo*	1.77	0.73	13
		$QSet_1$	2.85	1.07	
		$QSet_2$	2.54	1.27	
	UC	*Placebo*	1.62	1.12	13
		$QSet_1$	2.77	1.54	
		$QSet_2$	2.85	1.77	

6.6.1 Descriptive Statistics

The descriptive statistics from our experiments are summarised in Table 6.1, in Figure 6.4 on the next page and in Figure 6.5 on page 152, with appropriate rounding. Details are available in full in a supplementary technical report [88].

As seen in Figure 6.4 on the next page the mean initial contribution of written use cases was reasonably consistent among the cohorts, ranging from 0.8 at USYD to 1.2 at HU/b. The initial contribution of BPMN varied widely (from 0.4 at USYD to 1.4 at TU/e), and it was bigger than the initial contribution of written use case language for the participants from TU/e, but lower than written use cases at USYD. While the secondary contribution ($Cont_2$) of BPMN to TU/e was modestly positive, and big at USYD, the remaining secondary contribution ($Cont_2$)s were small or even negative. That is, the second workflow model did not contribute much, or worse, it confused participants, thus raising doubts of Green's view on cognitive

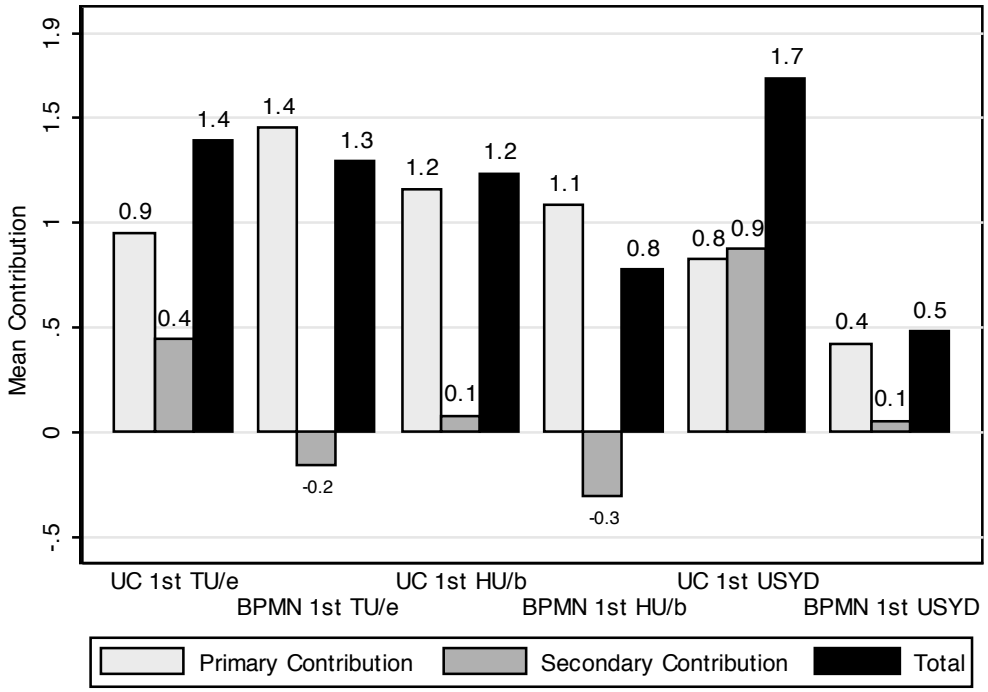


Figure 6.4 – Mean Contributions – All samples

dimensions [47] that suggests that presenting the same information twice, using different languages would always improve readability.

If we consider the combined impact of primary contribution ($Cont_1$) and secondary contribution ($Cont_2$), we see that presenting written use cases followed by BPMN gave consistent good results (from 1.2 to 1.7), while the reverse order had a wide variation (from 0.5 to 1.3).

6.6.2 Discussion of the Data

While the theoretical range of contributions was -6 to 6, the range of the average contributions we measured was between -3 and 5 (a negative contribution implied that the instrument confused the participant). For ten of the twelve tests we performed the average contributions were positive. The two exceptions were the TU/e and HU participants who received BPMN first. When subsequently presented with written use cases, their test results went down (see Figure 6.4 which compares the mean relative contribution of each language and of both languages to our six participant groups). In

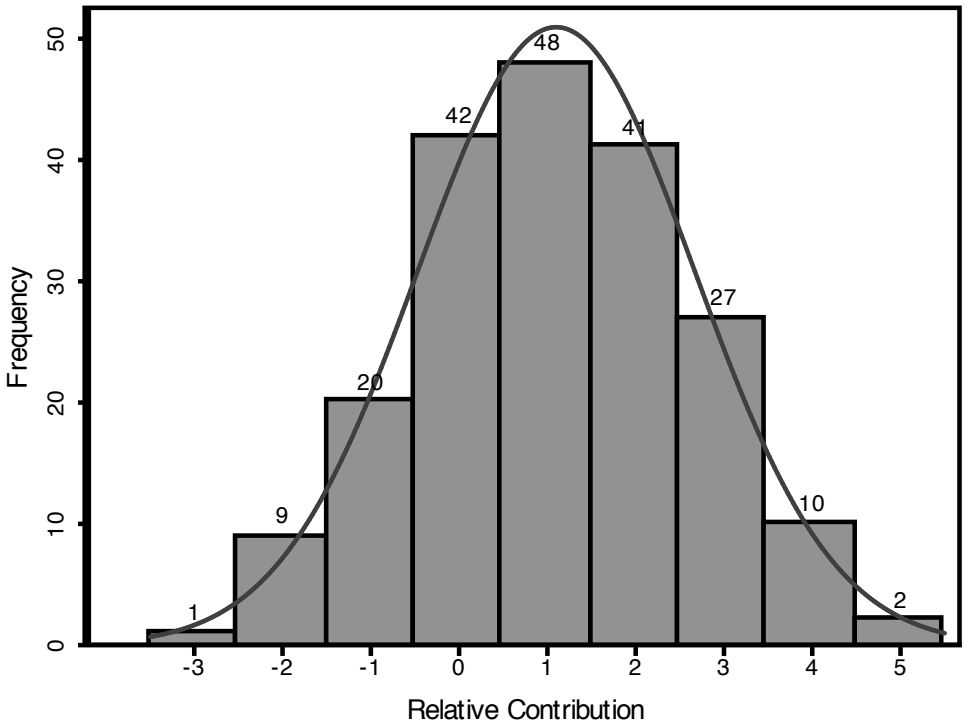


Figure 6.5 – Distribution of the Primary Contribution – All Participants – Both Languages

all the tests the initial domain knowledge (placebo) was bigger than the average contribution of any model.

Our findings suggest that the data are distributed normally. The Shapiro-Wilk W Test for Normal Data for $N < 5000$ [98] found that the test results are consistent with normality: $Placebo$ ($Prob > z = 0.7552$) $QSet_1$ ($Prob > z = 0.9958$) and $QSet_2$ ($Prob > z = 0.9982$).

6.6.3 Readability Comparison

To compare the two languages, we performed a range of common statistical tests on hypotheses 1 to 6. Table 2 shows the p-values from the one-sided Wilcoxon tests [113], for the two groups of participants: students at TU/e and HU/b, as a proxy for business analysts (BAs) and students at USYD, as a proxy for subject matter experts (SMEs). We use *italics* for p-values below 0.05. We also calculated other tests, such as t-tests; the details can

be found in [88] but the significant conclusions are the same.

There is support for H_1 (extended written use case language does contribute) and H_5 (extended written use case language followed by BPMN, makes more contribution than extended written use case language alone) at the statistically very significant 0.01 level, among both groups.

Had we considered only the business analysts proxies, we would have noticed strong support for H_2 (BPMN does contribute) and also support that is significant at 0.1 level for H_4 (BPMN contributes more than extended written use case language); however neither of these hypotheses seems well-supported for the subject matter expert proxies.

It is important for effective communication that a model be read correctly by diverse groups, including both business analysts and subject matter experts, so we see the data indicating that BPMN on its own is not a sufficient way to present process models, despite its success among those with training in process models.

Instead, our experiment offers reasons to provide a written presentation first, and then follow it with a graphical equivalent; this gave maximal scores among all the communities of readers.

6.6.4 Prediction

While searching for attributes that would help management predict which language is better for a particular team member, we asked participants to rate their comfort and experience with BPMN or written use case language using scales of one to five. We observed four OLS (ordinary least squares) linear regressions models as described below.

The model-estimates we provide are pairs of coefficients, one for the high scale and one for the low. We qualify each coefficient, in brackets, by a standard error, and a t-value. The coefficient reflects the weight of the variable and the impact of one unit change in the variable on the primary contribution ($Cont_1$) of the language. The standard error shows the variability in the weight calculated, it is the standard deviation of the variable divided by the square root of the sample size.

The t-value, which is the coefficient divided by the standard error, indicates significance for a two tailed test. If the absolute value of the t-value

Table 6.2 – P Values for One-Sided Wilcoxon Sign-Rank Tests

	BAs	**SMEs**	**Test**	Data Set 1	Data Set 2
H_1	*0.0000*	*0.0054*	within subjects	Primary Contribution(UC)	0
H_2	*0.0000*	0.1540	within subjects	Primary Contribution(BPMN)	0
H_3	0.9462	0.2352	between subjects	Primary Contribution(UC)	Primary Contribution(BPMN)
H_4	0.0542	0.7729	between subjects	Primary Contribution(BPMN)	Primary Contribution(UC)
H_5	*0.0066*	*0.0003*	within subjects	Secondary Contribution(BPMN)	0
H_6	0.9114	0.4521	within subjects	Secondary Contribution(UC)	0

is greater than 1.96, we have confidence that random chance is not likely to lead to the observed impact, so we say that the relevant attribute makes a significant difference in performance.

When asked if they were comfortable with flow charts, 82 participants responded that they strongly agree and one participant responded with strong disagreement. The remaining participants reported preferences somewhere in the middle. Participants at the higher end of the scale did not perform differently when compared to those at the lower end, with model estimates of −0.2258 (0.1507; −1.4980) for comfort, and 0.0642 (0.1576; 0.4075) for lack of comfort.

When asked if they were comfortable with the written use case languages, 73 participants responded that they strongly agree and 16 participants strongly disagreed. The remaining participants reported preferences somewhere in the middle. Participants at the higher end of the scale did not perform differently when compared to those at the lower end, with model estimates of −0.0823 (0.0986; −0.8350) for comfort and 0.0701 (0.1027; 0.6830) for lack of comfort.

When asked if they often worked with flow charts, 30 participants strongly agreed and six participant responded with strong disagreement. The remaining participants reported preferences somewhere in the middle. Participants at the higher end of the scale performed significantly better 0.2669 (0.1558; 2.3104). The performance of participants at the lower end varied −0.0865 (0.1351; −0.6404).

When asked if they often worked with the written use case language, 36 participants responded that they strongly agree and 20 participant responded with strong disagreement. The remaining participants reported preferences somewhere in the middle. Participants at the higher end of the scale performed significantly better at the 0.1 level 1.1875 (0.6426; 1.8479). The performance of participants at the lower end of the scale varied 0.1714 (0.3569; 0.4803).

We did find one statistically significant predictor. When asked how many fiction books they had read in the past 12 month, 73 participants reported none, 52 reported one to three, 26 reported four to six, 13 reported seven to ten, and 22 reported more than ten. *Our findings suggest*

a strong relationship between readership and textual aptitude. The range of the number of books that were read in the past 12 months ranged zero to more than ten. For the participants who first received written use cases as opposed to BPMN, the number of books they read were clustered via a Wards Hierarchical clustering routine, which revealed three distinct clusters of readership intensity. The first cluster comprised low readership, the third cluster comprised high readership levels. The primary contribution ($Cont_1$) of the extended written use case language was regressed over readership intensity. Low readership significantly predicted a negative effect on the primary contribution ($Cont_1$) with written use cases: -0.2370 (0.1125; -2.1070) and high readership predicted a significant positive effect: 0.4712 (0.1918; 2.4560). We also investigated this factor among the participants who received BPMN first: high readership predicted a significant negative effect on primary contribution ($Cont_1$) from the BPMN workflow model: -0.3911 (0.0867; -4.5092), but low readership was not a statistically significant predictor for success with BPMN: -0.1250 (0.1814; -0.6890).

6.6.5 Interpretation of the Results

The recruitment procedures we applied did not generate a random representative sample of the business analysts and business users in the industry. Nonetheless, there are some considerations that support the results to have external validity. Business users in a workplace had been students in a wide variety of fields, only a few years earlier. Both share the characteristic that they typically do neither have training in reading formal models nor modelling skills in general. The main difference that we would expect between students and business users is the level of initial domain knowledge, and we explicitly controlled for the effects of this in our analysis. Similarly, post-graduate students who study industrial engineering or business process modelling have been shown previously to be valid proxies for business analysts with low or median expertise in the industry [46, 91, 94]. These groups are both familiar with the concepts of formal models. Indeed, these cohorts of students demonstrated a good grasp of the particular notations in our experiment, with higher primary contribution from each notation than the generalist students from Sydney.

The recruitment procedures we applied did not generate a random representative sample of the business analysts and subject matter experts in the industry. None the less, due to the big size of the populations, the low p-values we found for some of the tests, and the strict control we exercised over the experimental procedure, we claim that some of our results can be generalised. In particular our findings suggest that participants from all groups can understand a model presented in written use cases, and business analysts can understand BPMN. Our results support H_1. Proxies for subject matter experts and for business analysts showed statistically significant increases in their understanding of models after reading a written use case presentation, when compared to their understanding using only background knowledge of the domain. In contrast, only proxies for business analysts showed statistically significant increases in their understanding of models after reading a BPMN presentation, when compared to their understanding from background knowledge of the domain.

Our findings show that participants from all groups who first read models employing the extended written use case language benefited further from the BPMN set. This was not true in the reverse order, indeed it even confused participants. Our results support H_5: proxies for subject matter experts and proxies for business analysts showed statistically significant increases in their understanding of a business problem from reading BPMN set following the delivery of written use cases. In contrast, the delivery of written use cases following the delivery of BPMN increased comprehension for subject matter experts but decreased it in business analysts. Dix *et al* state that formal specification should be accompanied by extensive commentary and a parallel sentential description [34, Page 596] and we agree.

We also suggest that the amount of fiction reading is predictive for the level of understanding of extended written use case language or BPMN. Other aspects of graphical or sentential aptitude, experience or preferences do not seem particularly informative.

6.7 Conclusion

We have taken care to craft a set of BPMN diagrams and written use cases that contain equivalent information. While doing so, we found that BPMN lacks an element of meta expressive power – the elements' identifiers which are presented in Cockburn's notation as the action steps labels, or as the extensions labels.

When we add a business reader to the equation, we leave Euclid's solid ground of slides and triangles and enter Shakespeare's shaky ground of intent and interpretations. That is where cognitive science and linguistics march in to our help.

Thus have we shown that in industrial context the extended written use case language is a more efficient communication device amongst stakeholders than the incumbent graphical languages.

6.8 Reflection

The hardest element of the research articulated in this chapter, and indeed in the entire research programme was the obtaining of an ethical approval. Accepting the "do no harm" principle as granted, another of the premises of the process is that students are to volunteer their services, or even to sell them. I have to question this assertion, as students benefit from older research whose results are incorporated into their syllabus, so I feel that they have the duty to generate research for their successors. I would thus call for a mandatory amount of hours a student should dedicate into research in proportion to the amount of research they consume. So a student who learns math, and consume no empirical research, will have to contribute less empirical researchers, than say, a medical student who consume lots of empirical research.

The approach I promote is that of partnership, between the experiment manager and the participants. That is the reason I have used the term "participants" rather then the more traditional term "subjects". In many ways I was disappointed that not one of the 210 participants in the experiment and the pilot study registered to receive the experiment's result even though they were all explicitly invited to do so.

Chapter 7

Conclusion

End of matter all was heard ...

Ecclesiastes 12:13

WORKFLOW CONFIGURATION OFFICERS construct formal process models from activities. As the workflow configuration officers rely on their past experience when constructing process models, they risk becoming detached from the working life. These process models guide workflow participants toward an expected result. Often, workflow participants use the process models as guides while enacting a business processes. Empowered workflow participants synthesise activities from the process models and from the conditions of the concrete situations. This synthesis forms a feedback loop in the course of a business process and becomes the basis for learning, which is embedded in each business process. This learning process, in turn, extends the process models. Deviations from process models are breakdowns, and therefore potential learning situations. Breakdown situations are all too common; they are a natural and very important part of any business process. Deviations should form the basis for learning and thus for developing and extending the process models.

In order for process models to become resources for the future realisation of activities, the review of the process models should be made as part each

activity. This calls for very readable workflow modelling languages. The understanding of process models as central dynamic assets of the organisation raises some unconventional requirements from workflow engines — instead of rigidly routing information around the organisations, the workflow engine should mediate the anticipation of the workflow configuration officers with recurrent events in working life. Hence, a workflow modelling language should support the modelling, verification, enactment, monitoring, and tuning of process models within a dynamic work environment.

The popular deployment of workflow engines, thus, necessitates more readable workflow modelling languages. The gap our research programme fills is that contemporary workflow modelling languages' sole function is to model workflow, they have the required expressive power but they are not an efficient interpersonal communication device. We find that readability by diverse groups of people is crucial for a workflow model to effectively bridge the various communities involved in the round trip of BPM — modelling, verification, enactment, monitoring, and tuning of workflow. Contemporary workflow modelling languages are not designed to produce artifacts that can be read, understood, reviewed, critiqued, and amended by workflow participants. Indeed workflow participants are not exposed to these models, and business analysts do not understand these models. We have not seen a workflow model expressed using contemporary workflow modelling languages, presented to workflow participants as a documentation of a process.

A candidate for a more approachable workflow modelling language shall be expressive, unambiguous, verifiable readable by wide a cohort of stakeholders, and acceptable by the industrial community.

Following industrial experience, having conjectured that the written use case language is worth using as a workflow modelling language, we undertook a research programme devoted to the examination of this conjecture. We broke our conjecture into five hypotheses which we presented, tested and assessed in this dissertation.

We started our journey by presenting in Chapter 2, a description of the state of the art, drawing on the achievements of three communities: business process management, requirements engineering and human computer

interaction. As our primary concern was languages we also drew upon work of the linguistics and the literacy communities.

Then, in Chapter 3 — Architecture, we provided a logical design of a use case oriented workflow engine. The design was detailed enough to enable a software developer to write a physical design of use case oriented workflow engine. This logical design was also sufficiently detailed for a workflow configuration officer to learn how to configure a use case oriented workflow engine, for a workflow participant to learn how to operate a use case oriented workflow engine, and for a business process manager to learn how to analyse the logs produced by the business activity monitor, a building block of a use case oriented workflow engine. Indeed, we articulated the use case oriented workflow engine idea by describing the architecture of such a workflow engine. In the process we highlighted and addressed some of the acknowledged limitations of Cocburn's written use case language.

We have then studied an implementation of a use case oriented workflow engine in the industry and learned that a use case oriented workflow engine was implemented in industrial settings and that the approach was well accepted by management, workflow configuration officers and workflow participants alike.

We then asked how expressive is our extended written use case language. We answered the question twice. In Section 5.2 — Patterns Approach, we demonstrated that *common* workflow patterns can be expressed using the written use case language. In Section 5.3 — Sound Workflow Nets Approach, we further generalised our investigation by providing a proof that *every reasonable* workflow pattern can be expressed using the written use case language. Then in section 5.4 — Unsafe Patterns, we provided three examples of use cases that are not safe, and called upon existing work to demonstrate how the safety violation can be identified.

Finally, in Chapter 6 — Readability, we assessed the readability of the written use case language and found that it is readable by BAs and workflow participants alike.

We now claim:

$Claim_1$**:** The written use case language can be used as a workflow modelling

language. We have provided a logical design of a use case oriented workflow engine and described in detail an industrial implementation of a use case oriented workflow engine.

Claim$_2$**:** The written use case language can express *common* workflow patterns. We have expressed a full library of workflow patterns as use case sets.

Claim$_3$**:** The written use case language can express every *reasonable* workflow pattern. We have proved that we can trace any sound workflow net by a use case set.

Claim$_4$**:** The written use case language can be tested for safety. We have shown this by calling upon Lee *et al* who have shown that use cases can be translated into Petri nets [66], which in tern one may test for safety by applying tools described by van der Aalst [3] and applied by Fahland *et al* [37].

Claim$_5$**:** The readability of the written use case language is higher than the readability of incumbent languages. We showed that the extended written use case language is an effective communication mechanism among heterogeneous stakeholders, which was not the case for BPMN.

Thus, having described the architecture of a use case oriented workflow engine, having described an industrial implementation of a use case oriented workflow engine, having proved that our extended written use case language is sufficiently rich to express every reasonable workflow patterns and having gathered statistically significant evidence that the extended written use case language is readable, we now suggest that the extended written use case language is worth using as a workflow modelling language.

Chapter 8

Further Research

> See the distress that we are in, how Jerusalem lies desolates , and her gates are burned with fire: come, and let us build up the walls of Jerusalem
>
> Nehemiah 2:17

The research programme on which we reported in this dissertation opened more questions than it closed. So it should.

The experiment, described in Chapter 6 assesses readability. It would be interesting to see how the various cohorts of participants react to demands for the changing of workflows.

The experiment, described in Chapter 6 assesses readability by students. It would be interesting to see how other cohorts of participants react to graphical or textual workflow modelling languages.

The use case oriented workflow engine approach was applied in commercial settings. It would be interesting to see if it can be applied to social networks, say to community managed workflows?

We measured a reduction in the comprehension of the participants from TU/e and HU/b, as participants' comprehension degraded after seeing BPMN followed by use cases. However statistically insignificant as it was, was this

an indication that we are not the rational being we would have liked to think we are?

But most of all, we did not attempt to come with a cognitive theory that explains our fundamental observations. Why is this that the extended written use case languages works well with untrained business people? A possible explanation may be that untrained business people best understand algorithms presented as a sequential sets of activities. The extended written use case language does a good job at breaking algorithms into sets of scenarios. These scenarios are uninterrupted sequences of activities that are clearly separated from their triggers. On the other hand, incumbent workflow modelling languages, for which BPMN is a proxy, mix activities with triggers as they break the sequence of activities while jumping between swim lanes. It is possible to design an experiment that will test this theory.

Appendices

Appendix A

Participant's workbook

The University of Sydney
ABN 15 211 513 464

School of Information Technologies
Faculty of Engineering
and Information Technologies

Associate Professor Alan Fekete
Room 447
School of IT, Building J12
University of Sydney NSW 2006
AUSTRALIA
Telephone: +61 2 9351 4287
Facsimile: +61 2 9351 3838
Email: fekete@it.usyd.edu.au

1 Participant's Information Statement

What is the study about?

We are doing some research in software engineering, a field which aims to improve the way software is developed. We are studying design notations that are used to communicate between IT professionals and the stakeholders such as managers for whom software is developed. In particular, we look at notations which describe the details of how a business process takes place. Different development teams use different notations when they explain this aspect to stakeholders; some of the notations are graphical and others use only text. While there has been a lot of research comparing the expressive power, accuracy and ambiguity of such notations, there is a gap, which we try to fill, in comparing them for usability and clarity. Notations that are not clear when read by stakeholders may cause unsatisfactory results, even if the notations are very expressive for the IT professional who writes the descriptions.

What does the study involve?

To conduct the research we have documented a business process using three different business process design notations. We imagine a fictitious company XYZ, and we document the process it uses to manage delivery of technological solutions to business problems. The process was written with one organisation in mind, so it is not generic, and probably does not fit many other organisations. As documenting a process is always a work in progress, we deliberately included some errors, omissions and inconsistencies. The inconsistencies and incompleteness simulate real life business modeling situations where designs are at times incomplete, ambiguous or even wrong.

In this experiment, you will be acting like a business stakeholder. You will be asked to read the descriptions of the business process, and we will explore how well the notation communicated to you. You have received a booklet that contains the experiment's material. As you read the material you will be asked three times to answer a short questionnaire. The questionnaires are identical. Our

experiment measures the difference between the answers, not the answers themselves. Please answer the questionnaires using all the information you know, rather than only the information we have given you most recently. Do not guess answers, if you do not know an answer to a question please tick the "I do not know" option. We have also included a "There is insufficient information" option. We are not assessing your skill, so you cannot be right or wrong; rather we are trying to judge the impact of the different design notations. Please read one section at a time and then answer its questionnaire. You may return to old sections.

Following all of this, we will ask you some questions to investigate your preference for communicating, whether you like graphical or verbal approaches.

Please do not write your name on the questionnaire, as we intend to keep the responses so they are not connected with your identity, in order to protect your privacy.

Who is carrying out the study?

The study is being conducted by PhD student Avner Ottensooser, and will form part of the basis for his thesis for the degree of Doctor of Philosophy at The University of Sydney under the supervision of Associate Professor Alan Fekete.

How much time will the study take?

The session will last approximately one hour. The session will consist of approximately 15 minutes for the background and post-experiment questionnaires, and approximately 45 minutes to complete the practical tasks.

Can I withdraw from the study?

Being in this study is completely voluntary - you are not under any obligation to participate. If you do consent to participate, you can later withdraw at any time until the booklet is handed in. Withdrawal will not affect your relationship with the researchers or the University of Sydney in any way. You may stop the experiment at any time if you do not wish to continue. However, once you have submitted your questionnaire you cannot withdraw as we will not be able to identify yours

Will anyone else know the results?

All aspects of the study, including individual results, will be strictly confidential even the researchers will not have access to information on participants. All recorded data and responses will not be associated with any name. While the envelope you received is numbered the booklet is not. A report of the study will be submitted for publication, but individual participants will not be identifiable in such a report.

Will the study benefit me?

You will receive AU$20.00 as compensation for your time. You are also making a valuable contribution to our ongoing research on design notations. The experience you have here may also be practice that can make your work easier if you pursue a career as manager or similar stakeholder who needs to interact with software development projects.

Can I tell other people about the study?

The study is not confidential and you are free to tell others.

What if I require further information?

If you are interested in the result of the research, of which you are now an important part, please drop us a note at `avner@it.usyd.nsw.au` . We will gladly share the research results with you in due course. When you have read this information, the lab supervisor will discuss it with you further and answer any questions you may have. If you would like to know more at any stage, please feel free to contact Avner Ottensooser (e-mail: `avner@it.usyd.edu.au`, ph: 0402 798 460), or Professor Alen Fekete (e-mail: `fekete@it.usyd.edu.au`, ph: 02 9351 4287).

What if I have a complaint or concerns?

Any person with concerns or complaints about the conduct of a research study can contact the Manager, Ethics Administration, University of Sydney on (02) 8627 8175 (Telephone); (02) 8627 8180 (Facsimile) or `gbriody@usyd.edu.au` (Email).

This information sheet is for you to keep

2 Instrument I – PARTICIPANT CONSENT FORM

This page is intentionally left empty.

The University of Sydney
ABN 15 211 513 464

School of Information Technologies
Faculty of Engineering
and Information Technologies

Associate Professor Alan Fekete
Room 447
School of IT, Building J12
University of Sydney NSW 2006
AUSTRALIA
Telephone: +61 2 9351 4287
Facsimile: +61 2 9351 3838
Email: fekete@it.usyd.edu.au

Participant's Consent Form

I, ...[PRINT NAME], give consent to my participation in the research project titled: *Process Notation – Usability Study.*

In giving my consent I acknowledge that:

1. I am at least 18 years old.

2. The procedures required for the project and the time involved have been explained to me, and any questions I have about the project have been answered to my satisfaction.

3. I have read the Participant Information Statement and have been given the opportunity to discuss the information and my involvement in the project with the researcher/s.

4. I understand that I can withdraw from the study at any time until I hand in the questionnaire, without affecting my relationship with the researcher(s) or the University of Sydney now or in the future. If I withdraw, my contact details will be removed from the files.

5. I understand that my involvement is strictly confidential and no information about me will be used in any way that reveals my identity. I understand that I may optionally give permission for my answered questioners and survey to be used in future research publications arising from this study.

6. I understand that being in this study is completely voluntary - I am not under any obligation to consent.

7. I consent to the use of my answered questionnaire and survey for research publications and presentations:

YES NO

Signed:.....

Name:.....

Date:.....

Process Notations Usability Studies

Avner OTTENSOOSER Alan FEKETE Hajo Reijers

September 9, 2009

3 Instrument II – Business Problem

With its global presence, company XYZ is the well established leader in the Australian financial services industry.

Company XYZ's management acknowledges that its core competency is not in the field of software development. However, due to the importance of software in the financial services industry, management had established a two tier strategy: usage of acquired software for general operations, and usage of in-house software innovations for the timely provision of solutions for core operations, so important for their competitive advantage.

Having witnessed several project failures, the company established a risk management culture. The company's management encourages the cancellation of projects that lag behind in cost justification measures, in order to free scarce resources.

Aiming at a repeatable process, the company instituted in 1998 a Solution Delivery Framework (SDF). The SDF articulates activities, deliverables and milestones mandated for the delivery of every software solution, whether acquired or developed in-house. The SDF's nature is that of a classical waterfall framework, gradually flowing between the development phases from requirement gathering, to specification, procurement, testing and deployment. Company XYZ adopted lessons from the contemporary iterative frameworks by concentrating on a multitude of small projects, close customer liaison, short development cycles and early testing, early delivery of pilots, and constant assessment and adjustment of upstream deliverables as solutions evolve.

To coordinate its investment in software, a project office had been established. Over the years the project office developed a spreadsheet based solution that supports the Solution Delivery Framework. The number and size of projects, let alone the complexity of supporting multiple teams have stretched the limits of the current solution and a new solution is required.

Acronym	Description	Description
SDF	Solution Delivery Framework	A process describing the activities that have to be performed, artifacts that have to be created and milestones that have to be met in order to deliver a software solution.
SSF	Strategic Solution Framework	A five year plan articulating the emerging business needs of company XYZ, solutions company XYZ will acquire to meet these needs, a time table, and build//buy classification.
RMP	Risk Management Profile	A description of the risks that could prevent success in the project, classified for seriousness, likelihood, ways of mitigating them, etc
SCD	Solution Concept Description	A very high level account of one way to provide the business with software that solves its problem
PSD	Project Scope Document	A description of the aspects that are covered by the solution, and those that are left to other projects
IR	Incident (bug) Report	A description of the circumstances in which the software did not behave as it was supposed to.
HLD	High Level Design	A high level description, showing the most important aspects only.
LLD	Low level design	The most detailed specification, issued to developers for coding.
TBA	To be advised	This acronym is used to acknowleged that an element in the design has not been decided yet.

Role	Description
Lead planner	A person who articulates all of the systems company XYZ will need over the planning horizon
Solution planner	A person who designs in detail one system that company XYZ will need in the short term
Project Manager	A person who has the overall responsibility for all resources needed to deliver one system that company XYZ needs in the short term.
Technical Lead	A person who has the overall responsibility for the technical aspects of one system that company XYZ needs in the short term
Project Board	A body that has the overall authority for the full development life-cycle of a solution
Project Office	A body that has the central authority for the full development life-cycle of all the solutions company XYZ has
Development Team	A body that has the responsibility to implement a single solution
Stake-holders	All people that will be effected by the system, directly or indirectly, whether they use it or not.

4 Instrument III -- Company XYZ Product offering

Research notes

This is the first design notation we evaluate. The philosophy of this design notation is that one must understand the products of a company to understand its processes.

Overview

Company XYZ has three families of products which it sells through three distribution channels.
The products are:

- A basket of real estate properties
- A family of indexed linked funds that represent equities and bonds in different markets.
- A book of loans issued to 23 of the S&P 500 companies.

These products are distributed through three channels:

- Managed funds issued to retail customers
- Retirement products issued to individuals either during the accumulation stage or the retirement stage
- A hedge fund issued by invitation only to selected high net worth individuals.

The XYZ real estate basket

With US$250 Billion under management company XYZ's controls basket of real estate properties ranging from the high speed rail (Beijing to Shanghai), airports (Denver, Frankfurt and Dubai), shopping malls and toll roads.

The XYZ family of index funds

The XYZ index funds aim to replicate the movements of an index of a specific financial market, or a set of rules of ownership that are held constant, regardless of market conditions. Tracking is achieved by trying to hold all of the securities in the index, in the same proportions as the index. Other methods include statistically sampling the market and holding "representative" securities. Some of company XYZ's index funds rely on a computer model with little or no human input in the decision as to which securities are purchased or sold and is therefore a form of passive management.
The lack of active management (stock picking and market timing) gives the

advantage of lower fees and lower taxes in taxable accounts. However, the fees will generally reduce the return to the investor relative to the index. Company XYZ finds that it is impossible to precisely mirror the index as the models for sampling and mirroring, by their nature, cannot be 100% accurate. The difference between the index performance and the fund performance is known as the 'tracking error' or informally 'jitter'.

The XYZ index funds track some common indices including the S&P 500, the Wilshire 5000, the FTSE 100 and the FTSE All-Share Index.

The XYZ loan book

A book of loans issued to 23 of the S&P 500 companies.

The XYZ wholesale funds

Managed funds issued to retail customers. The fund is liquid and customers who apply for units by 9:30 am will have same day value. Customers who request a redemption by 2pm will have the funds transfered to their bank accounts by close of business same day.

The XYZ retirement fund

Retirement products issued to individuals either during the accumulation stage or the retirement stage.

The XYZ Hedge fund

Company XYZ offers to wealthy individuals the opportunity to participate in a hedge fund that is designed to be profitable regardless of the directions of movements in the market. The fund is less liquid than the wholesale product.

The hedge funds are offered as either Capital Guaranteed flavor or naked flavor.

Capital Guaranteed The Capital Guaranteed flavor is a basket of high quality debt picked to mature at the instrument expiration date, and a basket of high beta long and short position, designed to achieve high return (albeit at high risk). The Capital Guaranteed product has a lifespan of five years during which redemptions are possible annually, at a high cost.

Naked The naked flavor is designed to have better performance over the medium range. Redemptions are processed every three months. Application for new units are currently closed.

5 Instrument IV – Comprehension Assessment Questionaire

Q1 What happens if stake-holders change the project scope?

1 The project stops

2 The project manager revisits the project plan

3 Scope creep is addressed in subsequent projects

4 The stake holders talk with developers who alter their code

5 I do not know

6 The handout does not supply information needed to answer the question.

7 None of the above.

Q2 Who reviews the SCD?

1 People nominated by the project office review the SCD

2 The Project Office review the SCD

3 The SCD is rarely reviewed

4 The project office reviews the SCD, and if the SCD is deemed valid, the project office nominates other approvers

5 I do not know

6 The handout does not supply information needed to answer the question.

7 None of the above.

Q3 What document does the PSD supplement?

1 The PSD supplements the SCD

2 The PSD is a core document? It supplements no other documents.

3 The PSD supplements the HLD and the LLD

4 None of the above

5 I do not know

6 The handout does not supply information needed to answer the question.

7 None of the above.

Q4 How many test cycles does the SDF allow?

1 One at the project end

2 As many as needed until no severe bugs are found or the project is cancelled.

3 Several, one at the end of every project stage

4 None - tests are not done in cycles but end users test all the time

5 I do not know

6 The handout does not supply information needed to answer the question.

7 None of the above.

Q5 What happens immediately after a solution is documented?

1 The testing team tests the solution

2 The customer tests the solution

3 The testing team writes detailed use-cases

4 The customer writes detailed use-cases

5 I do not know

6 The handout does not supply information needed to answer the question.

7 None of the above.

Q6 What condition determines when a build is reiterated?

1 Whether a severe bug has been found

2 Whether a not-severe bug has been found

3 Whether a severe bug has been fixed

4 Whether a build was completed

5 I do not know

6 The handout does not supply information needed to answer the question.

7 None of the above.

6 Instrument V – Solution expressed as written use cases

Research notes

This is another design notation we evaluate. The philosophy of this design notation is that words are good at managing complexity.

Use Case #0 Deliver a software solution

Use case Scope: High level

Trigger: General Manager nominates a lead planner.

Primary Actor: Lead Planner who wishes to establish a five year Solution acquisition plan.

Actors: Project team, Technical team, Project Manager, Operation control

```
1: Lead Planner creates Strategic Solution Framework (SSF).
2: Project team analyses requirements and designs a solution.
3: Technical team builds a solution and tests it.
4: Project manager performs ongoing management review.
5: Technical team deploys the solution to Operation Control.
6: Project Manager decommissions the project, retaining a skeleton maintenance team.
7: Operations Control conducts ongoing maintenance.
```

Figure 1. *Use Case #0 Deliver a software solution*

Use Case #1 Create a Solution Concept Document

Use case Scope: High level

Triggers: Lead Planner creates a Strategic Solution Framework, SSF, articulating core solutions and a five year project plan.

Risk management lead defines a company wide risk management framework, RMF, articulating project governance structures depending on Risk/Cost profile and adherence with project plan.

Brief description: A solution planner creates a Solution Concept Document

Primary Actor: A solution planner who wishes to plan a solution.

Actors: Project Office, Approvers, Development Lead, the Lead Planner

Minimal Guarantee: A strategy is written and reviewed .

```
1: A solution planner, concentrating on one solution in the SSF, creates a Solution
   Concept Document, SCD, articulating the current state, problems in the current
   states, requirements from a new solution, top down milestones and estimated
   benefits.                                                                  Order = 1

2: Project Office reviews the SCD.                                            Order = 2

3:  Project Office assigns approvers depending on the projects Risk/Cost profile.     Order = 3

4: Approvers review the SCD and approve the project.                          Order = 4

5: Development lead assigns a development slot.                               Order = 5

6: Project Office establishes a Solution Delivery Entry in the company's projects
   portfolio.                                                                 Order = 6

Extensions:

1a: The solution planner finds that the SCD cannot be implemented with current
      technology / resources with acceptable risk, time frame or cost.
      1a1:   The development of the solution stops.

2a:   Project office review finds the project concept document requires improvement
      2a1:   Technical lead repeats step 1.

2b:   Project office review finds the project's benefits to be insufficient or the
      project risk to be too high.
      2b1:   The project is cancelled.

4a:   Approver finds the project not worth pursuing.
      4a1:   The project is cancelled.

4b:   Approvers do not approve the project, specifying assumption errors.
      4b1:   The use case restarts at action step 1.

5a:   Resources are not unavailable.
      5a1:   The project is cancelled

5b:   The project will be possible only with less demand for resources
      5b1:   The use case restarts at action step 1.
```

Figure 2. *Use Case #1 Create a Solution Concept Document*

Use Case #2 Detail analysis and design

Use case Scope: High level

Trigger: Solution Concept Document (SCD) is completed and approved

Primary Actor: Project Manager who has to assess the project cost in detail

Actors: Project team, Technical Lead

```
Main success scenario

1: General Manager nominates a Project Manager.                                    Order = 1

2: Project Manager supplements the solution concept document with detail, writing a
   Project Scope Document (PSD).                                                  Order = 2

3: Project Manager assembles the project team, including direct reports, customers,
   suppliers and auditors.                                                         Order = 3

4: Project team reviews Project Concept Document.                                  Order = 4

5: Technical Lead writes High Level Design (HLD) evaluating several design avenues.  Order = 5

6: Technical Lead writes an assessment of technology and methods selecting one of
   the design alternatives.                                                        Order = 6

7: Technical lead conducts vendor selection.                                       Order = 7

8: Technical lead writes a Low Level Design (LLD) (Solution Design, Support
   impact, Risk assessment).                                                       Order = 8

9: Project manager writes Detailed Project Plan using input from the LLD (Time Line
   and Resources in Microsoft Project, Cash flow, Risk Management Plan).           Order = 9

Extensions:

2a:   The project manager finds the Solution Concept Document unrealistic
      2a1:   The project ends

4a:   The project team finds errors in the Project Concept Document
      4a1:   The use case starts at action step 2

8a:   While writing the LLD the technical lead finds errors in the HLD.
      8a1: The technical use case continues at action step 5.
```

Figure 3. *Use Case #2 Detail analysis and design*

Use Case #3.1 Build and test

Use case Scope: High level

Trigger: Low Level design completed

Description: This is the technical part of the project where the solution is crafted and verified.

Primary Actor: Technical Lead who wants to build the solution.

Actors: Development team, Testers, Project Manager, customers

```
1: Technical lead to write detailed use cases.                                    Order = 1

2: Technical lead conducts procurement.                                           Order = 2

3: Technical lead establishes development environments.                           Order = 3

4: Development team bring the solution to a build state.                          Order = 4

5: Development team to document the solution (Operational manual, training material,
   maintenance guide).                                                            Order = 5

6: Testers test the solution.                                                     Order = 6

7: Customers learn the solution and approve the solution.                         Order = 7


Extensions:

2a:   Technical lead finds that the negotiated costs invalidate the project
      benefits.
      2a1:   The use case is repeated from step 1.

2b:   Technical lead finds that the negotiated costs invalidates the project beyond
      repair.
      2b1:   The project is resumed from use case 2 step 4 (review project concept).

4a:   The development team identifies extensions not handled in the use cases
      4a1:   The use case continues at action step 1

4b:   The changes to the use case are significant
      4b1:   The Technical Lead refines the LLD
             (Reiterates use case 2 action steps 8 and 9).

6a:   Tester finds a bug and writes an IR
      6a1:   The project lead  finds the bug worth fixing
             6a1a:   The use case continues in step 4.

      6a2:   The technical lead finds the bug too minor to be fixed.
             6a2a:   The project continues in step 6.

      6a3:   The technical lead finds the bug and finds that the bug constitutes a change to the use cases.
             6a3a:   The use case continues from step 1.

      6a4:   The technical lead finds the bug a major scope change.
             6a4a:    The use case continues in use case 1 step 2 (back to the drawing board).
```

Figure 4. *Use Case #3.1 Build and test*

Use Case #3.2 Monitor Building

Use case Scope: High level

Primary Actor: The project manager who wants to track progress against plan.

Actor: Project Board, Project Office

```
1:   Project manager conducts communication with stake-holders                    Order = 1

2:   Project manager maintains and reviews risk log                               Order = 1

3:   Project manager monitors milestone adherence.                                Order = 1

4:   Project manager monitors budget compliance.                                  Order = 1

5:   Project manager writes a weekly project status report.                       Order = 2

6:   Project Board reviews project progress.                                      Order = 3

7:   The project office collates the minutes of the project board's meetings in the
     companys projects portfolio.                                                Order = 4


Extensions:

1a:   stake holders change the project scope
      1a1: TBA

3a:   New risk is identified
      3a1:   The project manager adds the new risk to the risk log

3b:   Existing risk becomes a reality
      3b1:   The project continues from use case 3.1 step 2.

3c:   Existing risk is rectified
      3c1:   The project manager erases the risk from the risk log

7a:   A project did not submit a weekly report
      7a1: TBA
```

Figure 5. *Use Case #3.2 Monitor Building*

Use Case #4 Deployment

Use case Scope: High level

Primary Actor: Development team that wants users to start using the system.

Actors: Users, Service Operation

```
1:   Development team writes a user manual.                                        Order = 1

2:   Development team conduct user training.                                       Order = 1

3:   Development team deploys the software.                                        Order = 2

4:   Development team hands over maintenance responsibility to Service Operations.  Order = 3

5:  After the end of the warranty period, development team ceases to maintain the
      project.                                                                     Order = 4

Extensions:

2a: New requirements are identified while training users
2a1:  The project resumes from use case 3.1 step 5.

2a: New bugs are identified while training users
2a1: The project resumes from use case 3.1 step 5.

5a: Too many bugs are found during the warranty period.
5a1: The Warranty period is extended.
```

Figure 6. *Use Case #4 Deployment*

Use Case #5 Close Project

Use case Scope: High level

Primary actor: Project manager who wishes to bring the project to closure.

Actors: Technical Manager, Solution Maintenance Team

```
1:   Technical Manager reviews vendor's contracts.                          Order = 1
2:   Project Manager conducts a post deployment review.                     Order = 2
3:   Project Manager conducts knowledge harvesting                          Order = 3
4:   Project Manager decommissions the project                              Order = 4
5:   Project Manager establishes a solution maintenance skeleton            Order = 5
6:   The project office removes the project from the project list           Order = 6
```

Figure 7. *Use Case #5 Close Project*

Use Case #6 Post deployment ongoing activities

```
Main Success scenario

1:   Operation management monitors the solution.                    Order = 1
2:   Help desk monitors users' activities.                          Order = 1
3:   Project manager reviews enhancement requests                   Order = 1
4:   Technical team develops enhancements / bug fixes               Order = 1
5:   Operations Management deploys enhancements                     Order = 1
```

Figure 8. *Use Case #6 Post deployment ongoing activities*

7 Instrument VI – Comprehension Assessment Questionaire

Q7 What happens if stake-holders change the project scope?

1 The project stops

2 The project manager revisits the project plan

3 Scope creep is addressed in subsequent projects

4 The stake holders talk with developers who alter their code

5 I do not know

6 The handout does not supply information needed to answer the question.

7 None of the above.

Q8 Who reviews the SCD?

1 People nominated by the project office review the SCD

2 The Project Office review the SCD

3 The SCD is rarely reviewed

4 The project office reviews the SCD, and if the SCD is deemed valid, the project office nominates other approvers

5 I do not know

6 The handout does not supply information needed to answer the question.

7 None of the above.

Q9 What document does the PSD supplement?

1 The PSD supplements the SCD

2 The PSD is a core document? It supplements no other documents.

3 The PSD supplements the HLD and the LLD

4 None of the above

5 I do not know

6 The handout does not supply information needed to answer the question.

7 None of the above.

Q10 How many test cycles does the SDF allow?

1 One at the project end

2 As many as needed until no severe bugs are found or the project is cancelled.

3 Several, one at the end of every project stage

4 None - tests are not done in cycles but end users test all the time

5 I do not know

6 The handout does not supply information needed to answer the question.

7 None of the above.

Q11 What happens immediately after a solution is documented?

1 The testing team tests the solution

2 The customer tests the solution

3 The testing team writes detailed use-cases

4 The customer writes detailed use-cases

5 I do not know

6 The handout does not supply information needed to answer the question.

7 None of the above.

Q12 What condition determines when a build is reiterated?

1 Whether a severe bug has been found

2 Whether a not-severe bug has been found

3 Whether a severe bug has been fixed

4 Whether a build was completed

5 I do not know

6 The handout does not supply information needed to answer the question.

7 None of the above.

8 Instrument VII – Solution expressed as BPMN style flow charts

Research notes

This is another design notation we evaluate. The philosophy of this design notation is that pictures are good at managing complexity (a picture is worth a thousand words).

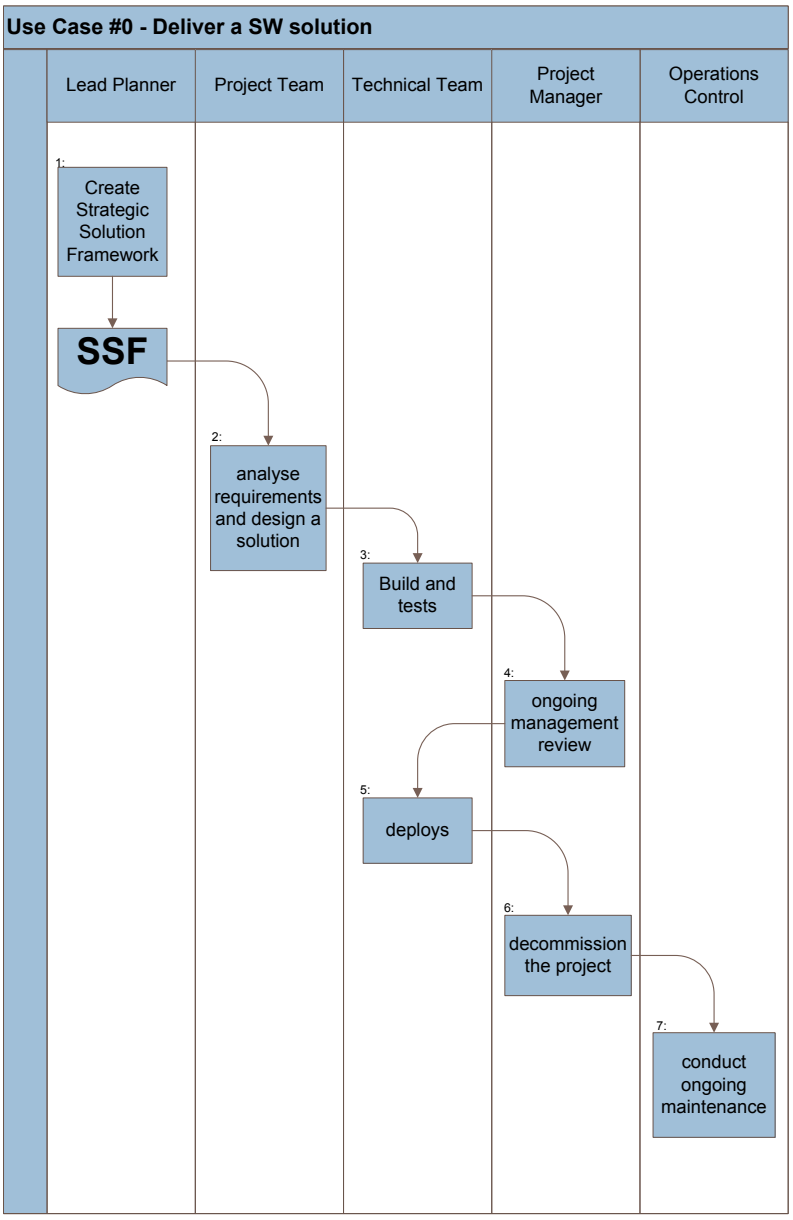


Figure 9. *Use Case #0 Deliver a SW solution*

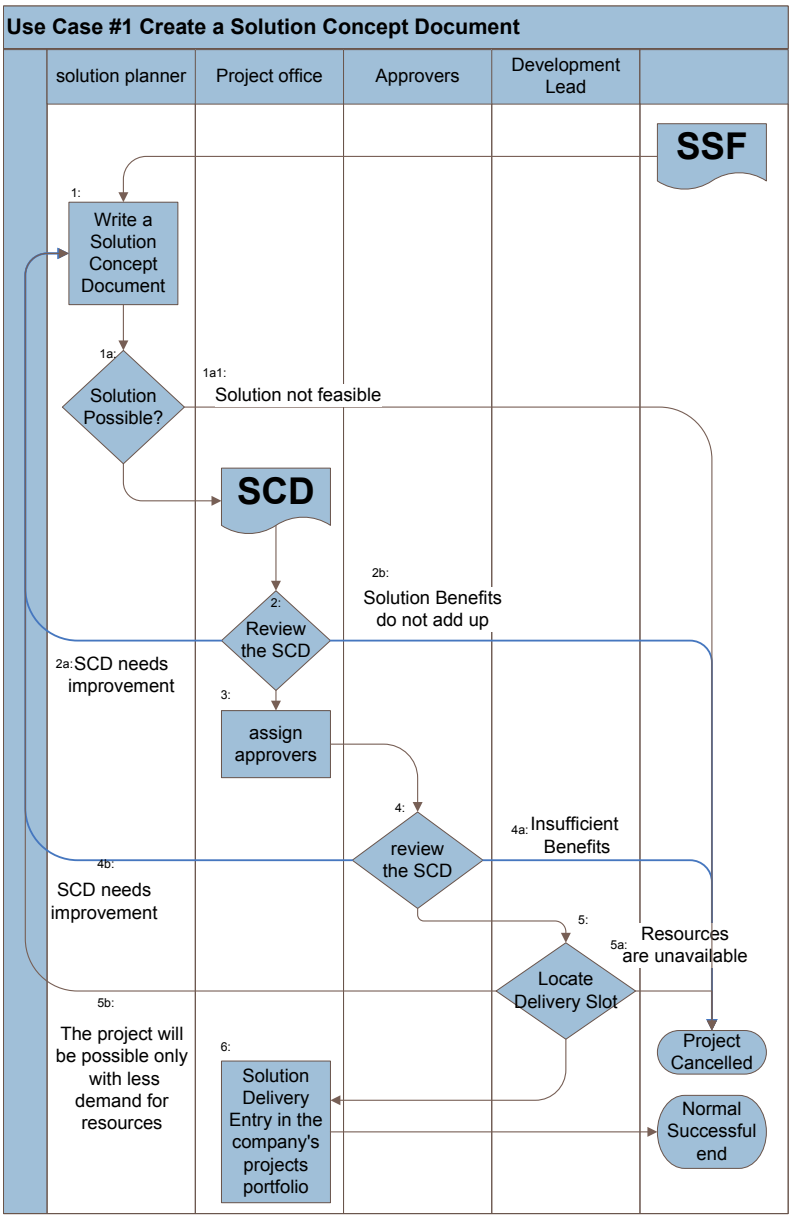


Figure 10. *Use Case #1 Create a Solution Concept Document*

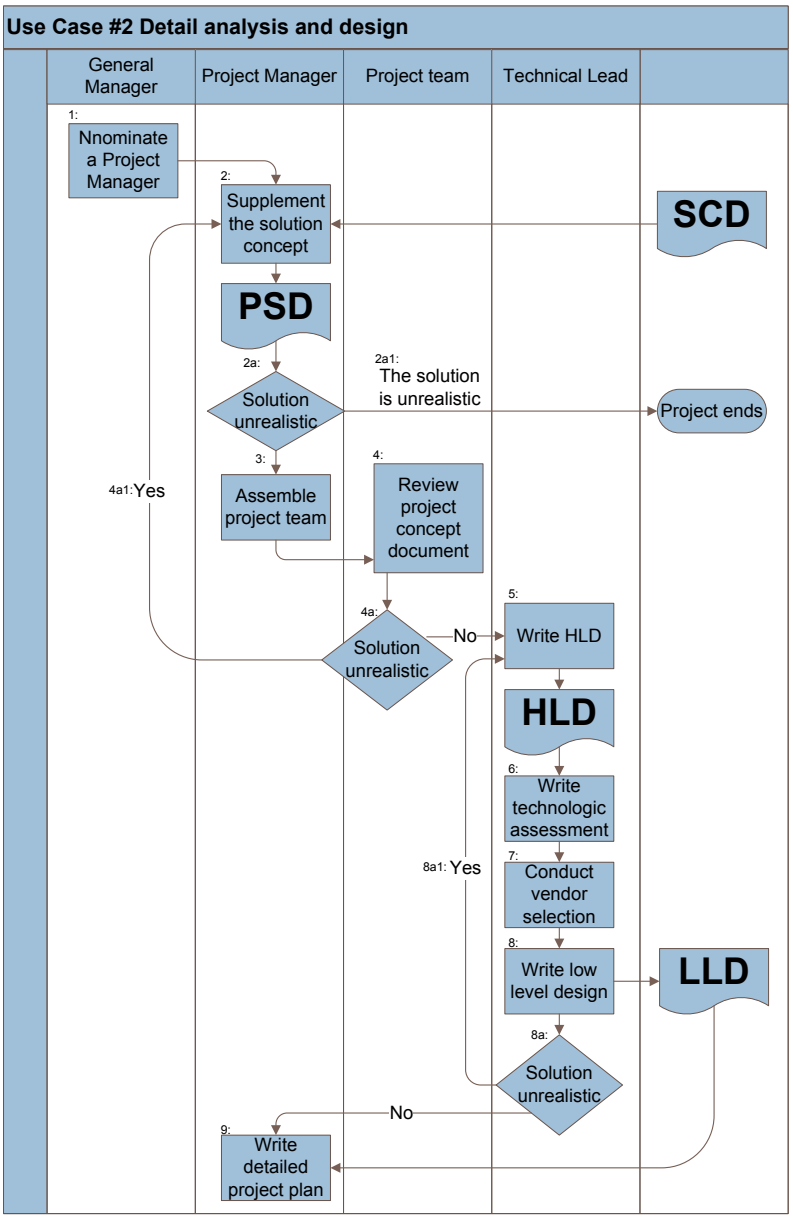


Figure 11. *Use Case #2 Detail analysis and design*

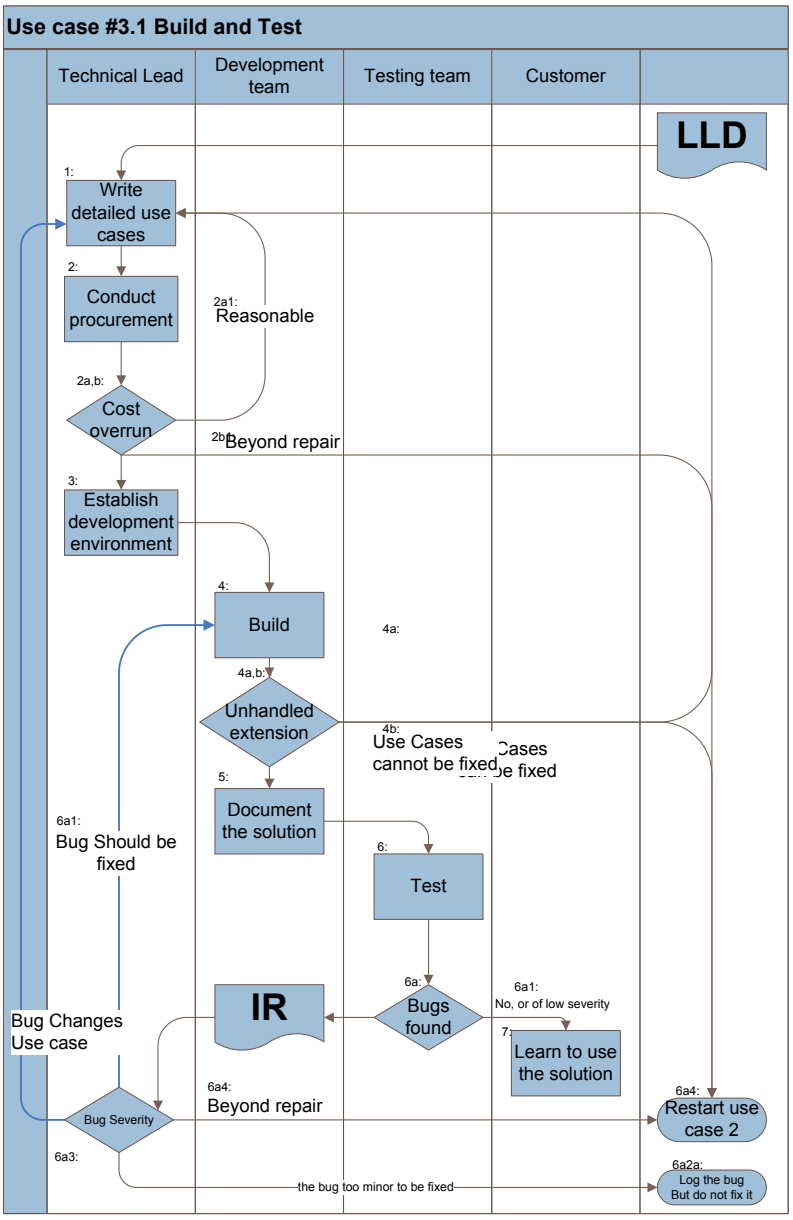


Figure 12. *Use case #3.1 Build and Test*

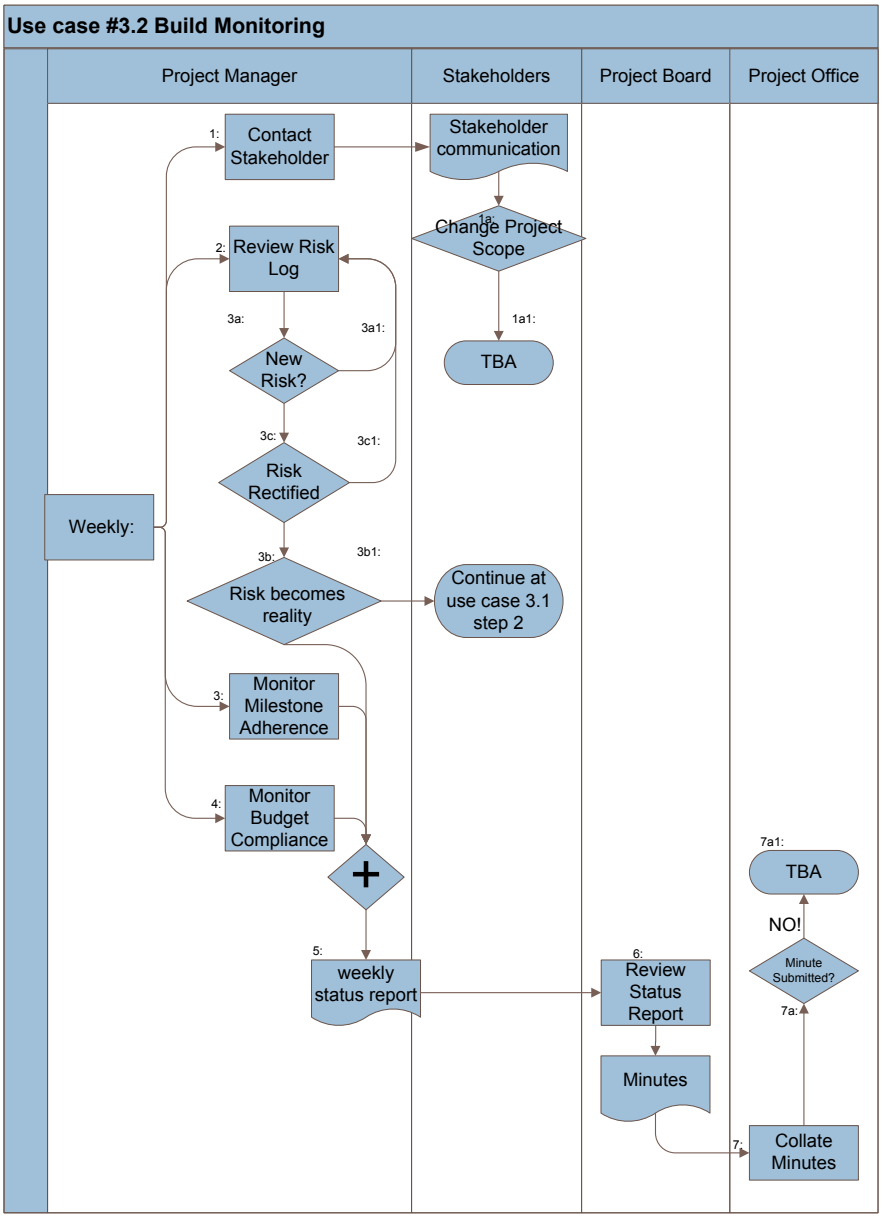


Figure 13. *Use case #3.2 Build Monitoring*

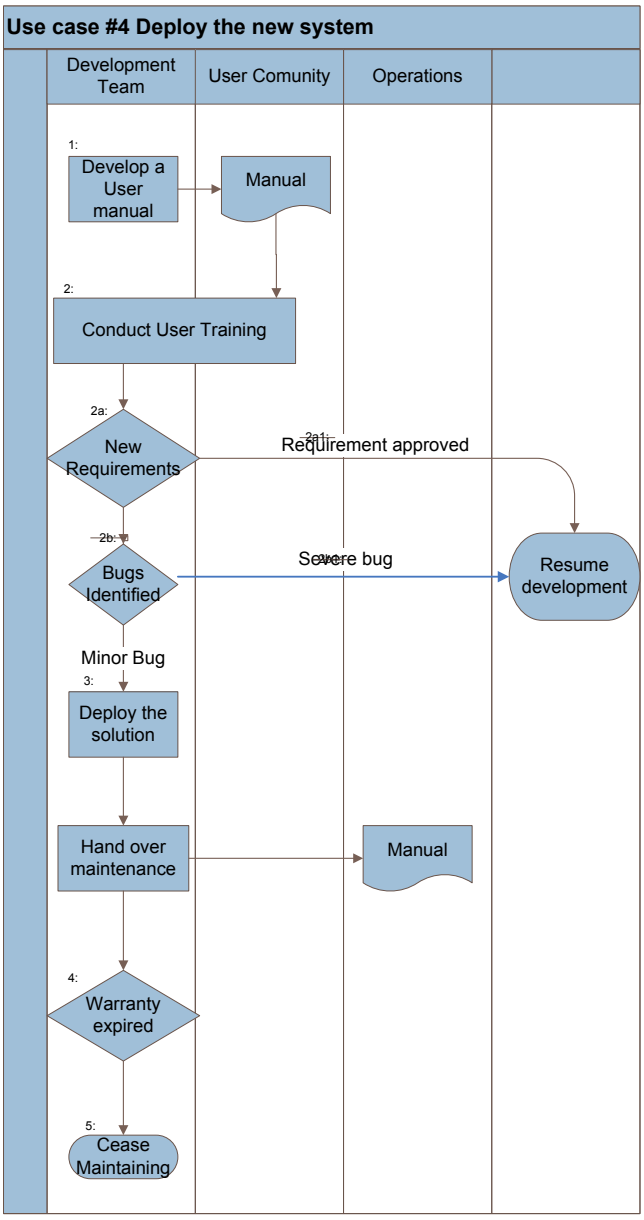


Figure 14. *Use case #4 Deploy the new system*

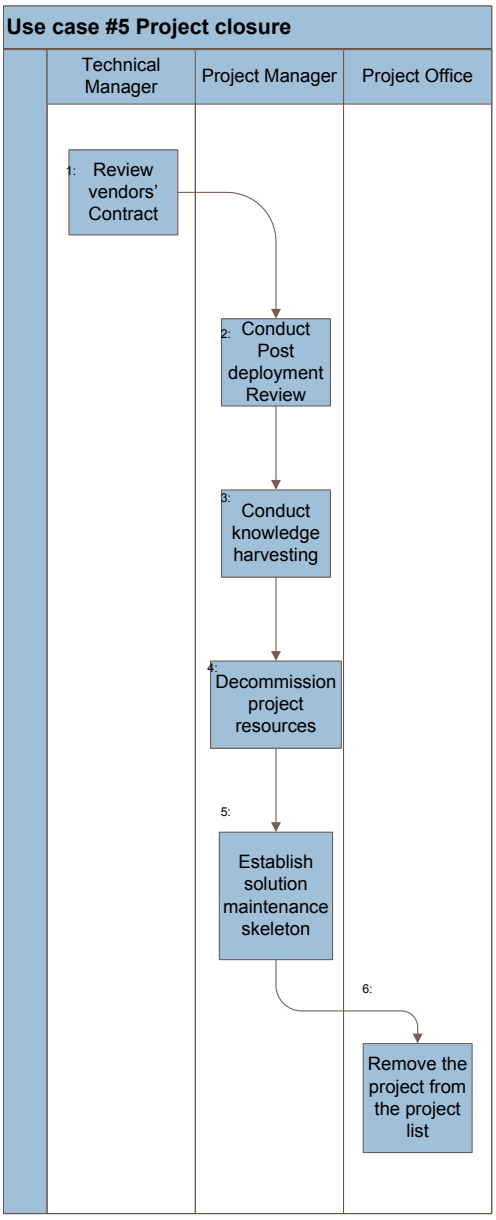


Figure 15. *Use case #5 Project closure*

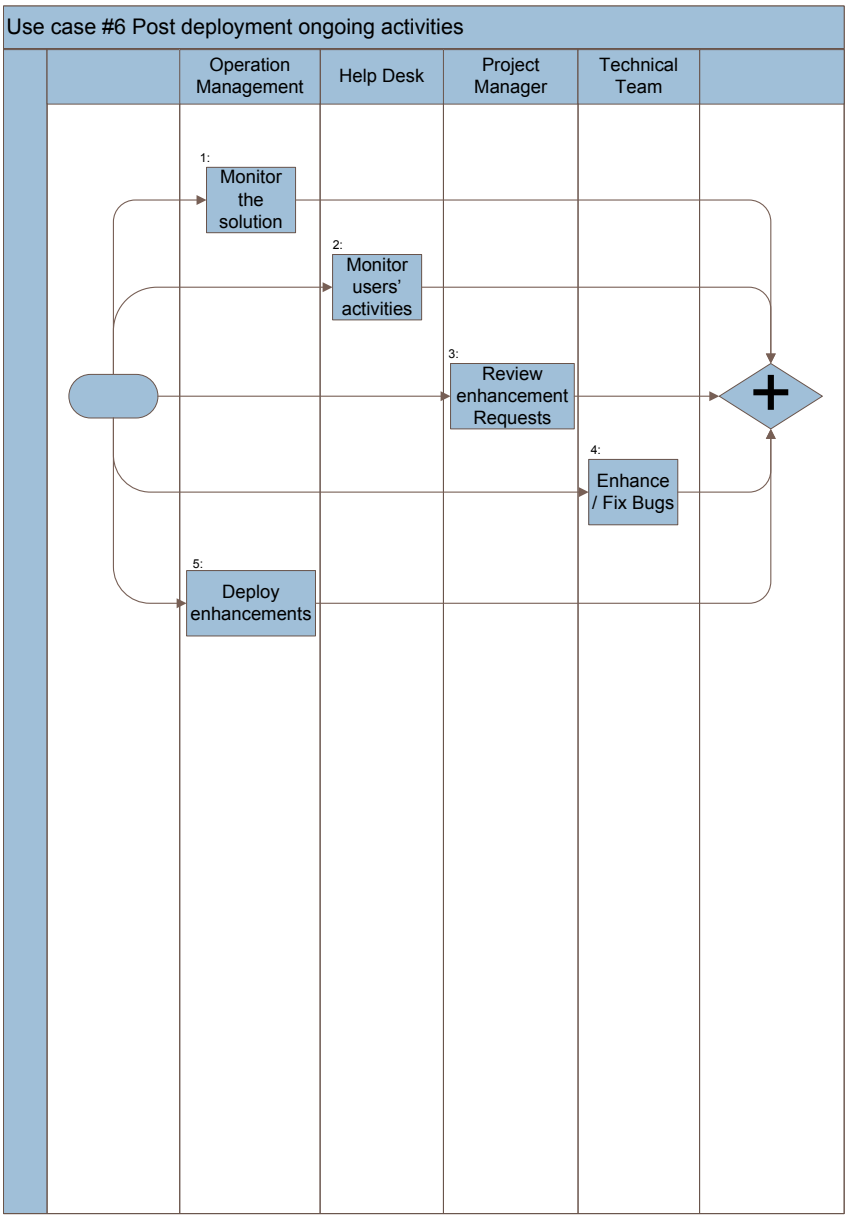


Figure 16. *Use case #6 Post deployment ongoing activities*

9 Instrument VIII – Comprehension Assessment Questionaire

Q13 What happens if stake-holders change the project scope?

1 The project stops

2 The project manager revisits the project plan

3 Scope creep is addressed in subsequent projects

4 The stake holders talk with developers who alter their code

5 I do not know

6 The handout does not supply information needed to answer the question.

7 None of the above.

Q14 Who reviews the SCD?

1 People nominated by the project office review the SCD

2 The Project Office review the SCD

3 The SCD is rarely reviewed

4 The project office reviews the SCD, and if the SCD is deemed valid, the project office nominates other approvers

5 I do not know

6 The handout does not supply information needed to answer the question.

7 None of the above.

Q15 What document does the PSD supplement?

1 The PSD supplements the SCD

2 The PSD is a core document? It supplements no other documents.

3 The PSD supplements the HLD and the LLD

4 None of the above

5 I do not know

6 The handout does not supply information needed to answer the question.

7 None of the above.

Q16 How many test cycles does the SDF allow?

1 One at the project end

2 As many as needed until no severe bugs are found or the project is cancelled.

3 Several, one at the end of every project stage

4 None - tests are not done in cycles but end users test all the time

5 I do not know

6 The handout does not supply information needed to answer the question.

7 None of the above.

Q17 What happens immediately after a solution is documented?

1 The testing team tests the solution

2 The customer tests the solution

3 The testing team writes detailed use-cases

4 The customer writes detailed use-cases

5 I do not know

6 The handout does not supply information needed to answer the question.

7 None of the above.

Q18 What condition determines when a build is reiterated?

1 Whether a severe bug has been found

2 Whether a not-severe bug has been found

3 Whether a severe bug has been fixed

4 Whether a build was completed

5 I do not know

6 The handout does not supply information needed to answer the question.

7 None of the above.

10 Instrument IX – Preference Survey

Q19 When asking for directions, would you prefer to receive:

A a map

B verbal instructions

C neither, or not applicable

Q20 When asking for directions, you most often receive:

A a map

B verbal instructions

C neither, or not applicable

Q21 When your team is asked to develop software, would you prefer:

A to design the solution yourself, and hand the design to a team member for coding.

B to receive a design from a team member, and code the solution yourself.

C neither, or not applicable

Q22 When your team is asked to develop software, most often you:

A design the solution yourself, and hand the design to a team member for coding.

B receive a design from a team member, and code the solution yourself.

C neither, or not applicable

Q23 The number of fiction books you read last 12 month is:

A 0

B 1-3

C 4-6

D 7-10

E more than 10

Q24 You are comfortable with flow charts:

A Strongly agree

B Somewhat agree

C I do not know

D Somewhat disagree

E Strongly disagree

Q25 You often work with flow charts:

A Strongly agree

B Somewhat agree

C I do not know

D Somewhat disagree

E Strongly disagree

Q26 You are comfortable with written use cases:

A Strongly agree

B Somewhat agree

C I do not know

D Somewhat disagree

E Strongly disagree

Q27 You often work with written use cases:

A Strongly agree

B Somewhat agree

C I do not know

D Somewhat disagree

E Strongly disagree

11 Instrument X – Demographic Survey

Q28 What is your gender?:

A female

B male

C I do not wish to answer this question

Q29 Which role best describes your occupation (Select all that apply)?

A student

B professional

C academic

D I do not wish to answer this question

Q30 Is English your native language?

A Yes.

B No.

C I do not wish to answer this question.

Q31 Do you have work experience related to Business Process Management and/or Modelling?

A Yes.

B No.

C I do not wish to answer this question.

Q32 Do you have any knowledge related to the software delivery life cycle?:

A Yes.

B No.

C I do not wish to answer this question.

Q33 Do you have any knowledge related to the financial industry?:

A Yes.

B No.

C I do not wish to answer this question.

Thank you for your participation

Appendix B

Pattern-Based Analysis of the Control-Flow Perspective of written-use-cases

Pattern WCP-01 (Sequence)

Description:

An activity in a workflow process is enabled after the completion of a preceding activity in the same process.

Examples

(a) The verify account activity executes after the credit card details have been captured.

(b) The codacil-signature activity follows the contract-signature activity.

(c) A receipt is printed after the train ticket is issued.

Use Case WCP-01a

```
Main success scenario:

1. Actor to capture credit card details.  Order = 1
2. Actor to verify account activity.      Order = 2
```

Figure B.1 – Use Case WCP-01a - Sequence

Pattern WCP-2 Parallel Split

Description:

The divergence of a branch into two or more parallel branches each of which execute concurrently.

Examples

(a) After completion of the capture enrolment activity, run the create student profile and issue enrolment confirmation activities simultaneously.

(b) When an intrusion alarm is received, trigger the despatch patrol activity and the inform police activity immediately.

(c) Once the customer has paid for the goods, issue a receipt and pack them for despatch.

Use Case WCP-02

```
Main success scenario:

1. Actor1 to capture enrolment.        Order = 1 ! Action step 1 is followed by an implicit AND-Split
2. Actor2 to create student profile.   Order = 2 ! Action steps 2 and 3 are parallel
3. Actor3 to issue enrolment activity. Order = 2
4. Process ends.                       Order = 3 ! AND-Join
```

Figure B.2 – Use Case WCP-02 — Parallel Split

Pattern WCP-3 Synchronisation (AND-Join)

Description:

The convergence of two or more branches into a single subsequent branch such that the thread of control is passed to the subsequent branch when all input branches have been enabled.

Examples

(a) The despatch-goods activity runs immediately after both the check-invoice and produce invoice activities are completed.

(b) Cash-drawer reconciliation can only occur when the store has been closed and the credit card summary has been printed.

Use Case WCP-03, covers examples a and b

```
Main success scenario:

1. Actor1 does A. Order = 1 ! Step 1 follows an implicit AND-Split
2. Actor2 does B. Order = 1 ! action-steps 1 and 2 are parallel
3. Actor3 does C. Order = 2 ! AND-Join
                            !
                            ! action-step 3 can start only after
                            ! the completion of both action-steps
                            ! 1 and 2.
```

Figure B.3 – Pattern WCP-03 Synchronisation (AND-Join)

Pattern WCP-4 Exclusive Choice

The divergence of a branch into two or more branches. When the incoming branch is enabled, the thread of control is immediately passed to precisely one of the outgoing branches based on the outcome of a logical expression associated with the branch.

Examples

(a) Depending on the volume of earth to be moved, either the dispatchbackhoe, despatch bobcat or despatch-D9-excavator activity is initiated to complete the job.

(b) After the review election activity is complete, either the declare results or the recount votes activity is undertaken.

Use Case WCP-4a

```
Main success scenario:
1. Validate that the volume of earth to be moved is small.   Order = 1
2. Dispatch backhoe.                                         Order = 2
3. End the process.                                          Order = 3   ! XOR-Join

Extension:

Action steps ‘2’, ‘1a1’, and ‘1b1’ are all not compatible. ! XOR


1a. The volume of earth to be moved is medium.             ! XOR-Split
    Replace action step 2 with:
    1a1. Dispatch bobcat                                     Order = 2

1b. The volume of earth to be moved is large               ! XOR-Split
    Replace action step 2 with:
    1b1. Dispatch D9-excavator                               Order = 2
```

Figure B.4 – Pattern WCP-4 Exclusive Choice — This approach complies with Cockburn Guideline 7: “Validate,” Don’t “Check Whether” [24, Page 95]. One may ask why don’t we place explicit restrictions on ‘1a’ and ‘1b’ as they are mutually exclusive. The answer is that our understanding of the process leads us to realise that of the two avenues, only one will be opened.

Pattern WCP-05 Simple Merge (XOR-Join)

Description:

The convergence of two or more branches into a single subsequent branch. Each enablement of an incoming branch results in the thread of control being passed to the subsequent branch.

Discussion:

The difference between WCP-3 and WCP-5 is that they are AND-Join and OR-Join respectively.

Examples

(a) At the conclusion of either the bobcat-excavation or the D9-excavation activities, an estimate of the amount of earth moved is made for billing purposes.

(b) After the cash-payment or provide-credit activities, initiate the produce-receipt activity.

Use Case WCP-5a

```
Main success scenario:
1 . Actor to validate that the amount of earth to be moved is small.  Order = 1
2.  Actor to despatch bobcat.                                         Order = 2
3.  Actor to bill                                                     Order = 3 ! XOR-Join

Extension:

1a. Amount of earth movement need is big                          ! XOR-Split
    Replace action step 2 with:
    1a1. Actor to despatch D9.                                      Order = 2
```

Figure B.5 – Pattern WCP-05 Simple Merge (XOR-Join) — This approach complies with Cockburn Guideline 7: "Validate," Don't "Check Whether" [24, Page 95]. Action steps '2' and '1a1' are mutually exclusive.

Pattern WCP-06 (Multi-Choice)

Description:

The divergence of a branch into two or more branches. When the incoming branch is enabled, the thread of control is passed to one or more of the outgoing branches based on the outcome of distinct logical expressions associated with each of the branches.

Example

Depending on the nature of the emergency call, one or more of the despatch-police, despatch-fire-engine and despatch ambulance activities is immediately initiated.

Use Case WCP-06a

```
Precondition:
Emergency call arrived.

Main success scenario:
1. Actor to asses call.                                          Order = 1
2. Processing to end.                                            Order = 3. ! OR-Join

Extension:

! Steps ‘1a’, ‘1b’ and ‘1c’ are compatible.
Step ’1d1’ is incompatible with step ’1a1’.     ! XOR
Step ’1d1’ is incompatible with step ’1b1’.     ! XOR
Step ’1d1’ is incompatible with step ’1c1’.     ! XOR

1a. Fire occurred.                              ! OR-Split
    1a1. Dispatch fire engine.                    Order = 2.

1b. Crime suspected                             ! OR-Split
    1b1. Dispatch Police.                         Order = 2.

1c. Injury occurred                             ! OR-Split
    1c1. Dispatch Ambulance.

1d. Fire and crime and injury did not occur     ! OR-Split
    1d1. TBA                                      Order - 2 ! The example does not state what
                                                            ! happens in that case, but implies that
                                                            ! something is sent.
```

Figure B.6 – Pattern WCP-06 (Multi-Choice) — This is the context most common in BT Financial Group as the customer makes arbitrary choice of features. If no routing sheet is prescribed - the work item is closed. The way the use case is written implies that it is possible not to send anything.

Pattern WCP-07 (Structured synchronising Merge)

Description:

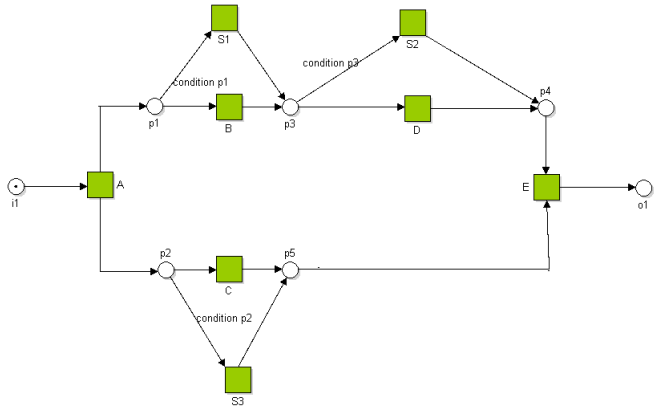
The convergence of two or more branches (which diverged earlier in the process at a uniquely identiable point) into a single subsequent branch. The thread of control is passed to the subsequent branch when each active incoming branch has been enabled.

Example

Depending on the type of emergency, either or both of the despatch-police and despatch ambulance activities are initiated simultaneously. When all emergency vehicles arrive at the accident, the transfer-patient activity commences.

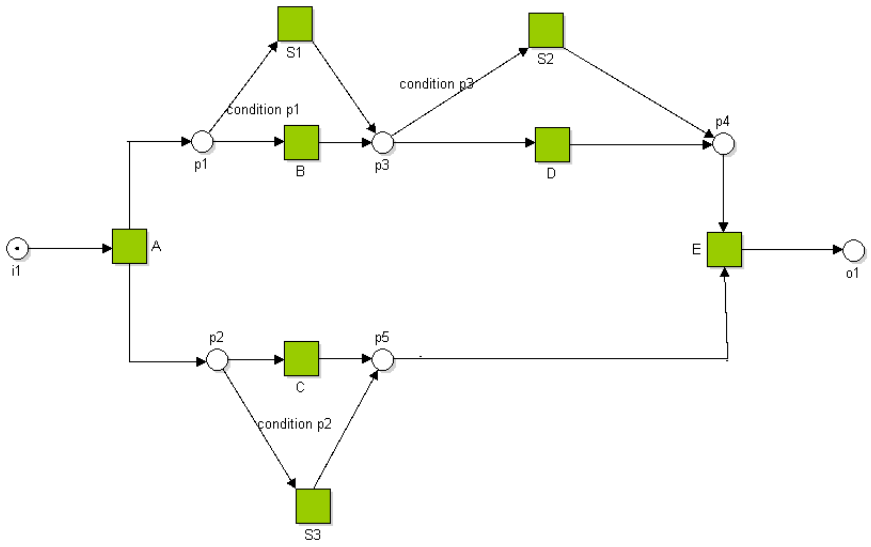
```
Use Case WCP-07a
Precondition:
Emergency call arrived.

Main success scenario:

1. Actor to assess damage.        Order = 1
2. Process ends.                  Order = 5

Extension:
! Action steps ‘2a’ and ‘2b’ can co-exist
Observations ‘2c’ and ‘2a’ are not compatible.
Observations ‘2c’ and ‘2b’ are not compatible.

2a. Crime is suspected          ! OR-Split
    2b1. Dispatch police.         Order = 2
    2b2. Police arrived.          Order = 3

2b. Injury occurred             ! OR-Split
    2b1. Dispatch ambulance.      Order = 2
    2b2. Ambulance arrives.       Order = 3
    2b3. Transfer patient         Order = 4 ! AND-Join

2c. Neither a crime is suspected nor an injury occurred
    2c1. TBA                      Order = 2 ! The example requires some thing to be despatched.
```

Figure B.7 – Pattern WCP-07a (Structured synchronising Merge)




```
Use Case WCP-07b
Main success scenario:
1. Actor does p2.              Order = 1
2. Actor does p3.              Order = 2
3. Actor does p4.              Order = 3
4. Actor does o1.              Order = 4  ! The Process ends

Alternative Scenario (Condition A)
1. Actor does p4.              Order = 1
2. Actor does o1.              Order = 2 ! The Process ends

Alternative Scenario (Condition B)
1. Actor does p1.              Order = 1
2. Actor does p5.              Order = 2
3. Actor does o1.              Order = 3 ! The Process ends

Alternative Scenario (Condition C)
1. Actor does p5.              Order = 1
2. Actor does o1.              Order = 2 ! The Process ends
```

Figure B.8 – Pattern WCP-07b (Structured synchronising Merge) In that case, the pilot prescribes one of the 4 scenarios. That example is illustrated using the Petri Net above




```
Use Case WCP-07c
Main success scenario (Condition A)
1. Actor does p2.                    Order = 1
2. Actor does p3.                    Order = 2
3. Actor does p4.                    Order = 3
4. Actor does o1.                    Order = 4  ! The process ends.

Alternative Flow (Condition B)
1. Actor does p1                     Order = 1
2. Actor does p5                     Order = 2
3. Actor does o1.                    Order = 3  ! The process ends.
}
```

Figure B.9 – WCP-7c — Structured synchronising Merge. That example is illustrated using the Petri Net above.

Pattern WCP-08 (Multi-Merge)

Description:

The convergence of two or more branches into a single subsequent branch. Each enablement of an incoming branch results in the thread of control being passed to the subsequent branch.

Example

The lay foundations, order materials and book labour activities occur in parallel as separate process branches. After each of them completes the quality review activity is run before that branch of the process completes.

```
Use Case WCP-08
Main success scenario
                                              ! Action steps 1, 2 and 3 follow and
                                              ! implicit AND-Split
1. Actor1 to lay foundation.       Order = 1 ! Action steps 1, 2 and 3 are parallel
2. Actor2 to order material.       Order = 1
3. Actor3 to book labourer.        Order = 1
4. Actor4 to review quality.       Order = 2 ! AND-Join
```

Figure B.10 – WCP-08 — (Multi-Merge)

Pattern WCP-9 (Structured discriminator)

Description:

The convergence of two or more branches into a single subsequent branch following a corresponding divergence earlier in the process model. The thread of control is passed to the subsequent branch when the first incoming branch has been enabled. Subsequent enablements of incoming branches do not result in the thread of control being passed on. The discriminator construct resets when all incoming branches have been enabled.

Example

When handling a cardiac arrest, the check breathing and check pulse activities run in parallel. Once the first of these has completed, the triage activity is commenced. Completion of the other activity is ignored and does not result in a second instance of the triage activity.

Use case WCP-9a

```
Precondition

Breathing check started.
Pulse check started.

Main success scenario

1. Actor to start the breathing test.                    Order = 1
2. Actor to start the pulse test.                        Order = 1
3. Actor to monitor the breathing test.                  Order = 2
4. Actor to monitor the pulse check.                     Order = 2
5. Process ends.                                         Order = 4 ! XOR-Join

Extension

Observations ‘3a’ and ‘4a’ are mutually exclusive       ! XOR

3a. Breathing test ended check ended first.             ! XOR-Split
    3a2. Actor to commence triage activity.               Order = 3

4a. Pulse check ended first.                            ! XOR-Split
    4a1. Actor to commence triage activity.               Order = 3
```

Figure B.11 – WCP-09a — (Pattern WCP-9 (Structured discriminator)

WCP-09 — Example b

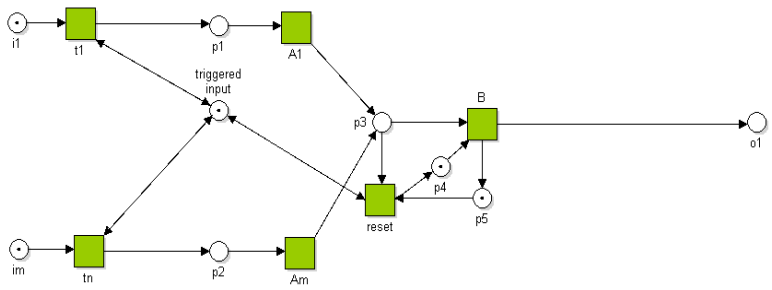
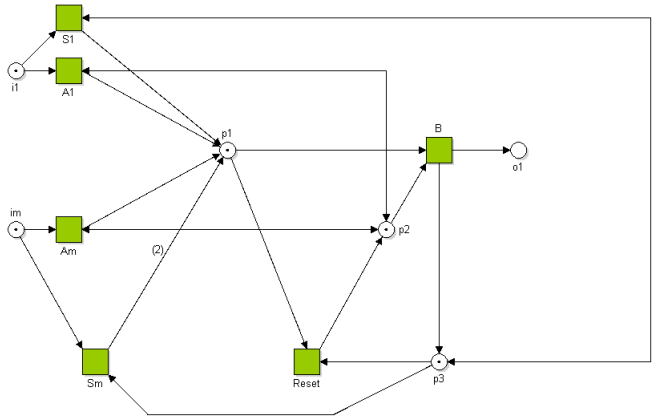


Figure B.12 – WCP-9b (Structured discriminator)

```
Main success scenario
1. Actor does P1          Order =  1
2. Process ends           Order =  4 ! XOR-Join

Extension
Action Steps ‘a1a’ and ‘1b’ are mutually exclusive. ‘1b’ can be prescribed several times. !XOR

1a. Condition B         ! XOR-Split
    12a1. Actor does P2   Order =  2

1b. Condition = D       ! XOR-Split
    1b1. actor does p3    Order =  2
    1b2. Actor does p2    Order =  3
```

Figure B.13 – WCP-09c — (Pattern WCP-9 (Structured discriminator)

WCP-09 — Example c

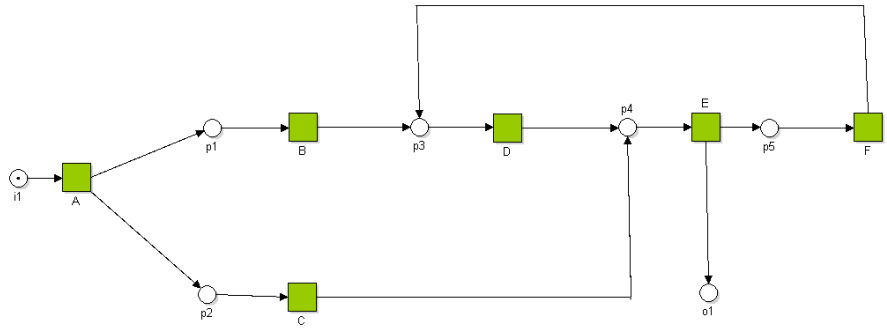



```
Main success scenario (Condition A)
! Routing sheet 2a and 2b are co-existing.
! Routing sheet 3a can be prescribed multiple times for one workflow.

1. Actor does p1.                          Order = 1
2. Actor does p3.                          Order = 2
3. Process ends.                           Order = 3

Alternative scenario 1 (Condition D)
4. Actor does p2.                          Order = 1
5. Actor does p3.                          Order = 2
6. Process ends.                           Order = 3

Alternative scenario 2 (Conditions (B or C))
7. Actor triggers input.                  Order = 1
8. Actor to redo the use case.            Order = 2

Extensions:

! Action steps 1a, 1b and 1c co-exist
2a. Condition G
    2a1. Actor does p5
    2a2. Actor does p4
    2a3. Actor does 01
    2a2a. Cond K.
          2a2a1. Actor does Trigger input.
5a. Condition M
    5a1 Actor does P4.
8a. Condition j
    8a1 Actor does Alternative scenario 1
8b. Condition L
    8b1 Actor does p4
```

Figure B.14 – WCP-09c — (Pattern WCP-9 (Structured discriminator)

Pattern WCP-10 (Arbitrary Cycles)

The ability to represent cycles in a process model that have more than one entry or exit point.




```
Use Case WCP-10

Main success scenario (Condition A)
    1. Actor does P1.                Order = 1
    2. Actor does P3.                Order = 2 ! OR-Join
    3. Actor does p4.                Order = 3
    4. Process ends.                 Order = 5 ! Step 4 is optional

Alternative scenario (Condition B)
    5. Actor does P4.                Order = 3
    6. Process ends.                 Order = 5

Extension
! Condition C can be observed multiple times.
*a  Condition C                    ! OR-Split
    *a1. Actor does p5.              Order = 4
    *a2. Actor does p3.              Order = 5
    *a3. Actor does p4.              Order = 6
    *a4. Process ends.               Order = 7
```

Figure B.15 – Pattern WCp-10 (Arbitrary Cycles) — The star notation indicate that the extension has multiple entry points ('3' and '5'), see [24, page 103].

Pattern WCP-11 (Implicit Termination) — Confidence - Trivial?

A given process (or sub-process) instance should terminate when there are no remaining work items that are able to be done either now or at any time in the future. The implicit termination feature is in the domain of the Workflow engine. Once all action-steps are tricked as done or as should not be done, the workflow item should be terminated.

Pattern WCP-12 (Multiple Instances without Synchronisation) — Confidence - Trivial?

Within a given process instance, multiple instances of an activity can be created. These instances are independent of each other and run concurrently. There is no requirement to synchronise them upon completion.

Example

A list of traffic infringements is received by the Transport Department. For each infringement on the list an Issue-Infringement-Notice activity is created. These activities run to completion in parallel and do not trigger any subsequent activities. They do not need to be synchronised at completion.

```
Use Case WCP-12a

Trigger: A traffic infringement row is received by the Transport Department.

Main success scenario

1. Actor to issue Infringement-Notice. Order = 1
```

Figure B.16 – Pattern WCP-12 (Multiple Instances without Synchronisation)

Pattern WCP-13 (Multiple Instances with a priory Design-Time Knowledge)

Description:

Within a given process instance, multiple instances of an activity can be created. The required number of instances is known at design time. These instances are independent of each other and run concurrently. It is necessary to synchronise the activity instances at completion before any subsequent activities can be triggered.

Example

The Annual Report must be signed by all six of the Directors before it can be issued.

```
Use Case WCP-13

Trigger

Annual report presented to signatures.

Main success scenario

1. Director to sign the annul report. Named User = A.    Order = 1
2. Director to sign the annul report. Named User = B.    Order = 1
3. Director to sign the annul report. Named User = C.    Order = 1
4. Director to sign the annul report. Named User = D.    Order = 1
5. Director to sign the annul report. Named User = E.    Order = 1
6. Director to sign the annul report. Named User = F.    Order = 1
7. Actor to issue the annual report.                     Order = 2  ! AND-Join
```

Figure B.17 – Pattern WCP-13 (Multiple Instances with a priory Design-Time Knowledge)

Pattern WCP-14 (Multiple Instances with a priory Run-Time Knowledge)

Within a given process instance, multiple instances of an activity can be created. The required number of instances may depend on a number of run time factors, including state data, resource availability and inter-process communications, but is known before the activity instances must be created. Once initiated, these instances are independent of each other and run concurrently. It is necessary to synchronise the instances at completion before any subsequent activities can be triggered.

Examples

(a) When diagnosing an engine fault, the check-sensor activity can run multiple times, depending on the number of error messages received. Only when all messages have been processed, can the identify-fault activity be initiated;

(b) In the review process for a journal paper submitted to a journal, the review paper activity is executed several times depending on the content of the paper, the availability of referees and the credentials of the authors. The review process can only continue when all reviews have been returned;

(c) When dispensing a prescription, the weigh compound activity must be completed

```
Use Case WCP-14a
Trigger: Diagnosing an engine fault requested

Main success scenario:
1. Actor to check-sensor .                    Order = 1
2. Process to end.                            Order = 3

Extension
*. Error found
add this scenario to the process:
    1a1. Actor to process fault.              Order = 2
```

Figure B.18 – Pattern WCP-14 (Multiple Instances with a priory Run-Time Knowledge) — The star notation indicate that the extension has multiple entry points (‘1’ and ‘1a2’), see [24, page 103].

Pattern WCP-15 (Multiple instances without a priory run-time knowledge)

Within a given process instance, multiple instances of an activity can be created. The required number of instances may depend on a number of run time factors, including state data, resource availability and inter-process communications and is not known until the final instance has completed. Once initiated, these instances are independent of each other and run concurrently. At any time, whilst instances are running, it is possible for additional instances to be initiated. It is necessary to synchronise the instances at completion before any subsequent activities can be triggered.

Examples

The despatch of an oil rig from factory to site involves numerous transport shipments activities. These occur concurrently and although sufficient activities are started to cover initial estimates of the required transport volumes, it is always possible for additional activities to be initiated if there is a shortfall in transportation requirements. Once the whole oil rig has been transported, and all transport shipment activities are complete, the next activity (assemble rig) can commence.

```
Use Case WCP-15
Main success scenario:

1. Actor to perform transportation activity.                    Order = 1
2. Actor to verify that all transportation completed.           Order = 3
3. Actor to assemble oil rig.                                   Order = 4  !OR-Join

Extension:

*a. More transportation required.                              ! OR-Split
    add this scenario to the process:
    *a1. Actor to perform additional transportation activity   Order = 2
```

Figure B.19 – Pattern WCP-15 (Multiple instances without a priory run-time knowledge) — This functionality is supported by our models ability to enable pilots to tailor the process after its start by prescribing action-steps on the fly.

Pattern WCP-16 (Deferred Choice)

A point in a workflow process where one of several branches is chosen based on interaction with the operating environment. Prior to the decision, all branches present possible future courses of execution. The decision is made by initiating the first activity in one of the branches i.e. there is no explicit choice but rather a race between different branches. After the decision is made, execution alternatives in branches other than the one selected are withdrawn

Examples

At the commencement of the Resolve complaint process, there is a deferred choice between the Initial customer contact activity and the Escalate to manager activity. The Initial customer contact is initiated when it is started by a customer services team member. The Escalate to manager activity commences 48 hours after the process instance commences. Once one of these activities is initiated, the other is withdrawn.

```
Use Case WCP-16
Main success scenario

1. Actor to start an issue.                          Order = 1
2. Actor to close issue.                             Order = 4  ! OR-Join

Extension



1a. Initial customer contact is initiated by
    customer service team.                         ! OR-Split
    Add this scenario to the use-aces:
    1a1. Customer service performs
         initial customer contact                    Order = 2

1b. 48 Hours passed
    Add this scenario to the use-aces:
    1b1. Manager performs escalate activity.         Order = 3  ! OR-Split
```

Figure B.20 – Pattern WCP-16 (Deferred Choice) — This is an example of mutual exclusive action-step

Pattern WCP-17 (Interleaved Parallel Routing)

A set of activities has a partial ordering denfing the requirements with respect to the order in which they must be executed. Each activity in the set must be executed once and they can be completed in any order that accords with the partial 34 order. However, as an additional requirement, no two activities can be executed at the same time (i.e. no two activities can be active for the same process instance at the same time).

Examples

When dispatching an order, the pick goods, pack goods and prepare invoice activities must be completed. The pick goods activity must be done before the pack goods activity. The prepare invoice activity can occur at any time. Only one of these activities can be done at any time for a given order.

The Work Flow item can be dispatch to one actor at a time.

```
Use Case WCP-17
Main success scenario:

1. Actor1 to pick goods. Order = 1
2. Actor1 to pack goods. Order = 2
3. Actor2 to invoice.    Order = Any
```

Figure B.21 – Pattern WCP-17 (Interleaved Parallel Routing)

Pattern WCP-18 (Milestone)

An activity is only enabled when the process instance (of which it is part) is in a specic state (typically in a parallel branch). The state is assumed to be a specific execution point (also known as a milestone) in the process model. When this execution point is reached the nominated activity can be enabled. If the process instance has progressed beyond this state, then the activity cannot be enabled now or at any future time (i.e. the deadline has expired). Note that the execution does not influence the state itself, i.e. unlike normal control-flow dependencies it is a test rather than a trigger.

Example

(a) Most budget airlines allow the routing of a booking to be changed providing the ticket has not been issued;

(b) The enrol student activity can only execute whilst new enrolments are being accepted. This is after the open enrolment activity has completed and before the close or enrolment activity commences.

```
Use Case WCP-18a
Trigger

Customer asked to change routing

Main success scenario

1. Actor accepts customer's call.                                    Order = 1
2. Actor issues ticket.                                              Order = 3
3. Customer to use the ticket.                                       Order = 5 ! OR-Join
3. End                                                               Order = 6

Extension

1a. Customer requests a routing change (before the ticket is issued)
    Add this scenario to the use-aces:
    1a1. Client contact to accept the request.                       Order = 2

2a. Customer requests a routing change (having been issued a ticket) ! OR-Split
    Add this scenario to the use-aces:
    2a1. Client-contact to (refuses, charges high amount).           Order = 4




Use Case 18b

Main success scenario:

1. Student to enrol.                                                   Order = 1

Extensions

1a. Student to request enrolment before opening day:                ! OR-Split
 Replace action step 1 with this scenario:
    ! that covers both before and after,
    ! as there is always a next term)

     1a1. System rejects enrolment.                                    Order = 1
     1a2. Student to be reminded of the upcoming enrolment period      Order = 2
```

Figure B.22 – Pattern WCP-18 (Milestone)

Pattern WCP-19 (Cancel Activity) and Pattern WCP-20 (Cancel Case)

Description:

The cancellation of activity can be done at any time using our model's "should not be done" action-step state. When all action-steps are either market as "Done" or "Should not be done", the dispatcher terminates the Workflow Item.

Examples

(a) The assess damage activity is undertaken by two insurance assessors. Once the first assessor has completed the activity, the second is cancelled;

(b) The purchaser can cancel their building inspection activity at any time before it commences.

```
Use Case 19a

Main success scenario:

1. Order assessment from assessor. Named user = a                    Order = 1
2. Order assessment from assessor. Named user = b                    Order = 1
3. Process claim.                                                    Order = 3

Extensions

*. First assessment arrived.
   add this scenario to the use-case:
    *1. Send assessment cancellation request to the other assessor. Order = 2
```

Figure B.23 – Pattern WCP-19 (Cancel Activity) — The star notation indicate that the extension has multiple entry points ('1' and '2'), see [24, page 103].

Pattern WCP-20 (Cancel Case)

Description:

A complete process instance is removed. This includes currently executing activities, those which may execute at some future time and all subprocesses. The process instance is recorded as having completed unsuccessfully.

Examples

(a) During an insurance claim process, it is discovered that the policy has expired and, as a consequence, all activities associated with the particular process instance are cancelled;

(b) During a mortgage application, the purchaser decides not to continue with a house purchase and withdraws the application.

Discussion:

This is an example where the"should not be done" structure is applicable.

Pattern WCP-21 (Structured Loop)

The ability to execute an activity or sub-process repeatedly. The loop has either a pre-test or post-test condition associated with it that is either evaluated at the beginning or end of the loop to determine whether it should continue. The looping structure has a single entry and exit point.

Example

(a) While the machine still has fuel remaining, continue with the production process.

(b) Only schedule fights if there is no storm activity.

(c) Continue processing photographs from the film until all of them have been printed.

(d) Repeat the select player activity until the entire team has been selected.

.

```
Use Case WCP-21a

1. Actor to verify that the machine has no fuel.          Order = 1
2. Actor to terminate the production proses.              Order = 4

Extension

*a. There is fuel.
    Supplement the use case with:
    *a1. Actor to continue with the production process.   Order = 2
    *a2. Actor to verify that the machine has no fuel.    Order = 3
```

Figure B.24 – Pattern WCP-21 (Structured Loop) — The above can be a action-step. That will solve the problem, but will create a misleading log. To enrich the log, a counter has to be added, so that the log can account for the number of iteration on the loop

Pattern WCP-22 (Recursion)

An instance of the resolve-defect activity is initiated for each mechanical problem that is identfied in the production plant. During the execution of the resolve-defect activity, if a mechanical fault is identied during investigations that is not related to the current defect, another instance of the resolve-defect is started. These sub-processes can also initiate further resolve-defect activities should they be necessary. The parent resolve-defect activity cannot complete until all child resolve-defect activities that it initiated have been satisfactorily completed.

Examples

An instance of the resolve-defect activity is initiated for each mechanical problem that is identified in the production plant. During the execution of the resolve-defect activity, if a mechanical fault is identified, during investigations, that is not related to the current defect, another instance of the resolve-defect is started. These sub-processes can also initiate further resolve-defect activities should they be necessary. The parent resolve-defect activity cannot complete until all child resolve-defect activities that it initiated have been satisfactorily completed.

```
Use Case WCP-22
Main success scenario

1. Actor to perform a clean verification                          Order = 1
2. Actor do declare the resolve defect satisfactorily completed.  Order = 4


Extensions

*a. A mechanical fault is identified
    *a1. Actor to resolve the fault.                              Order = 2
    *a2. Actor to perform a clean verification.                   Order = 3
```

Figure B.25 – Pattern WCP-22 (Recursion) — The above issue is fully resolved by my method by prescribing routing-sheets on the fly as issues are identified.

Pattern WCP-23 (Transient Trigger)

The ability for an activity to be triggered by a signal from another part of the process or from the external environment. These triggers are transient in nature and are lost if not acted on immediately by the receiving activity.

(a) Start the Handle Overflow activity immediately when the dam capacity full signal is received.

(b) If possible, initiate the Check Sensor activity each time an alarm trigger signal is received.

```
Use Case 23a
Main success scenario:

1. Observe the dam capacity                                  Order = 3

Extensions

Action-steps ‘*a1’ and ‘*b1’ are mutually exclusive.    ! XOR

*a. The dam capacity full signal is received.           ! Or-Split
    *a1. Start the Handle Overflow activity immediately   Order = 2
    *a2. Observe the dam capacity                         Order = 3

*b. The dam capacity full signal is off .               ! Or-Split
    *b1. Stop the Handle Overflow activity.               Order = 2
    *b2. Observe the dam capacity                         Order = 3
```

Figure B.26 – Pattern WCP-23 (Transient Trigger) — These are observation that trigger activities.

Pattern WCP-24 (Persistent Trigger)

The ability for an activity to be triggered by a signal from another part of the process or from the external environment. These triggers are persistent in form and are retained by the workflow until they can be acted on by the receiving activity.

Examples

(a) Initiate the Staff Induction activity each time a new staff member event occurs.

(b) Start a new instance of the Inspect Vehicle activity for each service overdue signal that is received.

```
Use Case WCP-24a and b
Main success scenario:

1. Observe.                                          Order = 2


Extensions

*a. A new staff member event occurred.
    *a1. Initiate the staff induction activity.  Order = 1
    *a2. Observe.                                Order = 2
```

Figure B.27 – Pattern WCP-24 (Persistent Trigger) — These are observation that trigger action-steps.

Pattern WCP-25 (Cancel Region)

The ability to disable a set of activities in a process instance. If any of the activities are already executing, then they are withdrawn. The activities need not be a connected subset of the overall process model.

Examples

(a) Stop any activities in the Prosecution process, which access the evidence database from running.

(b) Withdraw all activities in the Waybill Booking process after the freight-lodgment activity.

Discussion:

The model supports this using selective cancellation of multiple action-steps. A single click activity on a cluster of action-steps can be implemented.

Pattern WCP-26 (Cancel Multiple Instance Activity)

Within a given process instance, multiple instances of an activity can be created. The required number of instances is known at design time. These instances are independent of each other and run concurrently. At any time, the multiple instance activity can be cancelled and any instances which have not completed are withdrawn. This does not aect activity instances that have already completed.

Example

Run 500 instances of the Protein Test activity with distinct samples. If it has not completed one hour after commencement, cancel it.

```
Use Case 26a
Main success scenario

1. Actor to run 500 instances of the protein Test activity with distinct samples. Order = 3
2. Actor to do some thing with the test result.                                   Order = 4

Extension

1a. An hour elapsed and the tests were not completed !
    1a1. Actor to cancel the tests.                                               Order = 1
    1a2. Process terminated.                                                      Order = 2
```

Figure B.28 – Pattern WCP-26 (Cancel Multiple Instance Activity)

Pattern WCP-27 (Complete Multiple Instance Activity)

The required number of instances is known at design time. These instances are independent of each other and run concurrently. It is necessary to synchronize the instances at completion before any subsequent activities can be triggered. During the course of execution, it is possible that the activity needs to be forcibly completed such that any remaining instances are withdrawn and the thread of control is passed to subsequent activities.

Example

Run 500 instances of the Protein Test activity with distinct samples. One hour after commencement, withdraw all remaining instances and initiate the next activity.

```
Use case 27

Main success scenario

1. Actor to run 500 instances of the protein Test activity with distinct samples. Order = 1
2. Actor to do some thing with the test result                                    Order = 2

Extension

1a. An hour elapsed and the tests were not completed
    1a1. Actor to cancel the remaining tests.                                     Order = 3
```

Figure B.29 – Pattern WCP-27 (Complete Multiple Instance Activity)

Pattern WCP-28 (Blocking Discriminator)

The convergence of two or more branches into a single subsequent branch following one or more corresponding divergences earlier in the process model. The thread of control is passed to the subsequent branch when the rst active incoming branch has been enabled. The discriminator construct resets when all active incoming branches have been enabled once for the same process instance. Subsequent enablements of incoming branches are blocked until the discriminator has reset.

Example

When the first member of the visiting delegation arrives, the check credentials activity can commence. It concludes when all delegation members have arrived. Owing to staff constraints, only one instance of the check credentials activity can be undertaken at any time. Should members of another delegation arrive, the checking of their credentials is delayed until the first check credentials activity has completed.

```
Use Case 28
Trigger

First member of the visiting delegation arrives

Main success scenario

1. Actor to check credentials.                          Order = 1


Extension

1a. More members of the current delegation arrive
    1a1. Repeat step 1                                  Order = 1

1b. Members of another delegation arrive
    1b1 Actor to ask them to wait                       Order = 1
```

Figure B.30 – Pattern WCP-28 (Blocking Discriminator) — This is an assignment of work to a single processor to a single queue until all of the action-steps are completed.

Pattern WCP-29 (Cancelling Discriminator)

The convergence of two or more branches into a single subsequent branch following one or more corresponding divergences earlier in the process model. The thread of control is passed to the subsequent branch when the first active incoming branch has been enabled. Triggering the discriminator also cancels the execution of all of the other incoming branches and resets the construct

Example

After the extract-sample activity has completed, parts of the sample are sent to three distinct laboratories for examination. Once the first of these laboratories completes the sampleanalysis, the other two activity instances are cancelled and the review-drilling activity commences.

```
Use Case 29
Main success scenario

1. Actor to extract sample            Order = 1
2. Actor sends sample to labs A-C     Order = 2
3. Wait                               Order = 5

Extensions

routing sheets 2a, 2b and 2c are mutualy exclusive.

2a. Lab A returned sample first      ! OR-Split
    2a1. Actor to cancel labs B and C. Order = 3
    2a2. Actor to review drilling.     Order = 4


2b. Lab B returned sample first      ! OR-Split
    2b1. Actor to cancel labs A and C. Order = 3
    2b2. Actor to review drilling.     Order = 4

2c. Lab C returned sample first      ! OR-Split
    2c1. Actor to cancel labs A and B. Order = 3
    2c2. Actor to review drilling.     Order = 4
```

Figure B.31 – Pattern WCP-29 (Cancelling Discriminator)

Pattern WCP-30 (Structured Partial Join)

The convergence of M branches into a single subsequent branch following a corresponding divergence earlier in the process model. The thread of control is passed to the subsequent branch when N of the incoming branches have been enabled. Subsequent enablements of incoming branches do not result in the thread of control being passed on. The join construct resets when all active incoming branches have been enabled.

Example

Once two of the preceding three Expenditure Approval activities have completed, trigger the Issue cheque activity. Wait until the remaining activities have completed before allowing the Issue Cheque activity to fire again.

```
Use Case 30
1. Actor to process Expenditure Approval              Order = 1
2. Actor to process Expenditure Approval              Order = 1
3. Actor to trigger cheque activity                   Order = 1
4. Actor to process remaining Expenditure Approval    Order = 1
5. Actor to fire Issue Cheque activity                Order = 2
```

Figure B.32 – Pattern WCP-30 (Structured Partial Join)

Pattern WCP-31 (Blocking Partial Join)

The convergence of two or more branches into a single subsequent branch following one or more corresponding divergences earlier in the process model. The thread of control is passed to the subsequent branch when N of the incoming branches have been enabled. The join construct resets when all active incoming branches have been enabled once for the same process instance. Subsequent enable- ments of incoming branches are blocked until the join has reset.

Example

When the first member of the visiting delegation arrives, the check credentials activity can commence. It concludes when 80 percent of delegation members have arrived. Owing to staff constraints, only one instance of the check credentials activity can be undertaken at any time. Should members of another delegation arrive, the checking of their credentials is delayed until the first check credentials activity has completed.

```
1. Actor to process members of the visiting delegation. Order = 2

Extension
1a. 80% of delegation members have arrived.          ! Or-Split
    1a1. Actor to end processing delegation.           Order = 1

1b. members of another delegation arrive             ! Or-Split
    1b1. Inform of delay.                              Order = 3
    1b2. Place member at the new queue.                Order = 4
```

Figure B.33 – Pattern WCP-31 (Blocking Partial Join) — This is implemented by instructing a single processor to serve a queue until 80% of delegation member has arrived and then serve a second logical queue.

Pattern WCP-32 (Cancelling Partial Join)

The convergence of two or more branches into a single subsequent branch following one or more corresponding divergences earlier in the process model. The thread of control is passed to the subsequent branch when N of the incoming branches have been enabled. Triggering the join also cancels the execution of all of the other incoming branches and resets the construct.

Example

Once the picture is received, it [or an image of it AO] is sent to three art dealers for the examination. Once two of the prepare condition report activities have been completed, the remaining prepare condition report activity is cancelled and the plan restoration activity commences.

```
Use Case 32
Main success scenario

1. Actor to send picture to dealer A                  Order = 1
2. Actor to send picture to dealer B                  Order = 1
3. Actor to send picture to dealer C                  Order = 1
4. Plan restoration activity                          Order = 2

Extensions

4a. Both dealers A and B returned the picture       ! OR-Split
    4a1. Actor to cancel dealer-Cs examination       Order = 3


4b. Both dealers A and C returned the picture       ! OR-Split
    4b1. Actor to cancel dealerbs examination        Order = 3

4c. Both dealers B and C returned the picture       ! OR-Split
    4c1. Actor to cancel dealeras examination        Order = 3
```

Figure B.34 – Pattern WCP-32 (Cancelling Partial Join)

Pattern WCP-33 (Generalised AND-Join)

The convergence of two or more branches into a single subsequent branch such that the thread of control is passed to the subsequent branch when all input branches have been enabled. Additional triggers received on one or more branches between firing of the join persist and are retained for future rings.

Examples

(a) When all Get Directors Signature activities have completed, run the Complete Contract activity.

(b) Accumulate engine, chassis and body components from the various production lines. When one of each has been received, use one of each component to assemble the basic car.

```
Use Case 33a
Main success scenario

1. Get Director signatures.                    Order = 1
2. Actor to run the Complete Contract activity. Order = 2
```

Figure B.35 – Pattern WCP-33 (Generalised AND-Join)

Pattern WCP-34 (Static Partial Join for Multiple Instances)

Within a given process instance, multiple concurrent instances of an activity can be created. The required number of instances is known when the first activity instance commences. Once N of the activity instances have completed, the next activity in the process is triggered. Subsequent completion of the remaining $M - N$ instances are inconsequential.

Example

Examine 10 samples from the production line for defects. Continue with the next activity when 7 of these examinations have been completed.

```
Use Case 34a
Main success scenario

1.  Examine 10 samples from the production line for defects  Order = 2

Extention

1a. 7 of these examinations have been completed            ! OR-Split
    1a1. Actor to continue with the next activity           Order = 1
```

Figure B.36 – Pattern WCP-34 (Static Partial Join for Multiple Instances)

Pattern WCP-35 (Cancelling Partial Join for Multiple Instances)

Description:

Within a given process instance, multiple concurrent instances of an activity can be created. The required number of instances is known when the first activity instance commences. Once N of the activity instances have completed, the next activity in the process is triggered and the remaining M-N instances are cancelled.

Example

Run 500 instances of the Protein Test activity with distinct samples. Once 400 of these have completed, cancel the remaining instances and initiate the next activity.

```
Use Case 35
Main success scenario

1. Actor to Run 500 instances of the Protein
   Test activity with distinct samples.            Order = 3

   ! note that activity one will be executed before 1a1 as the condition was not raised.

Extension

1a. 400 of these have completed.                 ! OR-Split
    1a1. Actor to cancel the remaining instances   Order = 1
    1a2. Actor to initiate the next activity       Order = 2
```

Figure B.37 – Pattern WCP-35 (Cancelling Partial Join for Multiple Instances)

Pattern WCP-36 (Dynamic Partial Join for Multiple Instances)

Within a given process instance, multiple concurrent instances of an activity can be created. The required number of instances may depend on a number of runtime factors, including state data, resource availability and inter-process communications and is not known until the initial instance has completed. At any time, whilst instances are running, it is possible for additional instances to be initiated providing the ability to do so has not been disabled. A completion condition is spec ied which is evaluated each time an instance of the activity completes. Once the completion condition evaluates to true, the next activity in the process is triggered. Subsequent completions of the remaining activity instances are inconsequential and no new instances can be created.

Examples

The despatch of an oilrig from factory to site involves numerous transport shipment activities. These occur concurrently and although sufficient activities are started to cover initial estimates of the required transport volumes, it is always possible for additional activities to be initiated if there is a shortfall in transportation requirements. Once 90 percent of the transport shipment activities are complete, the next activity (invoice transport costs) can commence. The remaining transport shipment activities continue until the whole rig has been transported.

```
Use Case 36a
Main success scenario
1. Actor to order 90% of transport             Order = 1
2. Actor to invoice transport costs            Order = 2
3. Actor to continue to order transport.       Order = 3

Use Case 36b
Main success scenario
1. Actor to order transport                    Order = 2
Extensions
1a. 90\% of transport was ordered            ! OR-Split
    1a1  Actor to invoice transport costs      Order = 1
```

Figure B.38 – Pattern WCP-36 (Dynamic Partial Join for Multiple Instances)

Pattern WCP-37 (Acyclic Synchronizing Merge)

Description:

The convergence of two or more branches which diverged earlier in the process into a single subsequent branch. The thread of control is passed to the subsequent branch when each active incoming branch has been enabled. Determination of how many branches require synchronization is made on the basis of information locally available to the merge construct. This may be communicated directly to the 69 merge by the preceding diverging construct or alternatively it can be determined on the basis of local data such as the threads of control arriving at the merge.

Example

```
Main success scenario

1. Actor to do P1                    Order = 1
2. Actor to do P3                    Order = 2
3. Actor to do p4                    Order = 3
4. Process ends                      Order = 4 !XOR merge

Extension

Action steps (2 and 3) as a group are incompatible with action steps (1b1 and  1b2) as a group

1a. Condition B was raised         ! XOR Split
    Replace action steps 2 and 3 with:
    1b1. Actor to do P2              Order = 2
    1b2. Actor to do P5              Order = 3
```

Figure B.39 – Pattern WCP-37 (Acyclic Synchronizing Merge)

Pattern WCP-38 (General Synchronizing Merge)

The convergence of two or more branches which diverged earlier in the process into a single subsequent branch. The thread of control is passed to the subsequent branch when each active incoming branch has been enabled or it is not possible that the branch will be enabled at any future time.

Example

```
Use Case WCP-38
Main success scenario

1. Actor to do P1               Order = 1
2. Actor to do P3               Order = 2
3. Actor to do p4               Order = 3
4. Process ends                 Order = 4

Extension

Action steps (2 and 3) are together incompatible with action steps (1b1 and 1b2) together.

1a. Condition B was raised
   Replace action steps 2 and 3 with:
   1b1. Actor to do P2         Order = 2
   1b2. Actor to do P5         Order = 3


4a. Condition C was raised
    4a1. Actor to repeat the main success scenario
```

Figure B.40 – Pattern WCP-38 (General Synchronizing Merge)

Pattern WCP-39 (Critical Section)

Two or more connected sub graphs of a process model are identified as “critical sections”. At runtime for a given process instance, only activities in one of these “critical sections” can be active at any given time. Once execution of the activities in one “critical section” commences, it must complete before another ”critical section” can commence.

Example

Both the take-deposit and final-payment activities in the holiday booking process require the exclusive use of the credit-card-processing machine. Consequently only one of them can execute at any given time.

```
Use Case WCP-39
Main success scenario

1. Actor to take-deposit           order = 1
2. Actor to process final payment  order = 2
```

Figure B.41 – Pattern WCP-39 (Critical Section) — The uniqueness of the credit card processing machine should be implemented as queue served by only one actor.

Pattern WCP-40 (Interleaved Routing)

Each member of a set of activities must be executed once. They can be executed in any order but no two activities can be executed at the same time (i.e. no two activities can be active for the same process instance at the same time). Once all of the activities have completed, the next activity in the process can be initiated.

Example

The check-oil, test-feeder, examine-main-unit and review-warranty activities all need to be undertaken as part of the machine-service process. Only one of them can be undertaken at a time, however they can be executed in any order.

```
Use Case WCP-40
Main success scenario

1. Actor to check-oil          Order = 1
2. Actor to test-feeder        Order = 1
3. Actor to review-warranty    Order = 1

  ! The uniqueness is maintained by the
  ! dispatcher and the unique supporting document.
```

Figure B.42 – Pattern WCP-40 (Interleaved Routing)

Pattern WCP-41 (Thread Merge)

At a given point in a process, a nominated number of execution threads in a single branch of the same process instance should be merged together into a single thread of execution.

Example

Instances of the register-vehicle activity run independently of each other and of other activities in the Process Enquiry process. They are created as needed. When ten of them have completed, the process-registrationbatch activity should execute once to finalise the vehicle registration system records update.

```
Use Case WCP-41
Main success scenario

1. Actor to register-vehicle                          Order = 2

Extension

1a. Ten vehicle registrations have been completed  ! AND-Split
    1a1. process-registrationbatch activity          Order = 1
```

Figure B.43 – Pattern WCP-41 (Thread Merge)

Pattern WCP-42 (Thread Split)

At a given point in a process, a nominated number of execution threads can be initiated in a single branch of the same process instance.

Example

At the completion of the confirm paper receival activity, initiate three instances of the subsequent independent peer review activity.

```
Use Case WCP-42

Trigger

Paper arrival

Main success scenario:

1. Actor to confirm paper revival      Order = 1
2. Acror to send paper to reviewer 1   Order = 2 ! AND-Split
3. Actor to send paper to reviewer 2   Order = 2 ! AND-Split
4. Actor to send paper to reviewer 3   Order = 2 ! AND-Split
```

Figure B.44 – Pattern WCP-42 (Thread Split)

Pattern WCP-43 (Explicit Termination)

A given process (or sub-process) instance should terminate when it reaches a nominated state. Typically this is denoted by a specific end node. When this end node is reached, any remaining work in the process instance is cancelled and the overall process instance is recorded as having completed successfully.

Example

N/A

```
Use Case WCP-43
Main success scenario

1. Actor to do A                Order = 1
2. Actor to do B                Order = 2
3. Actor to do C                Order = 3
4. Actor to do D                Order = 4
5. Terminate the process        Order = 5

Extension

2a. Something was observed     ! OR-Split
    2a1. Terminate the process   Order = 5
```

Figure B.45 – Pattern WCP-43 (Explicit Termination)

B.1 Conclusion

This completes our demonstration that written Use-Cases, with the extra notation for order and coexistence, are richly expressive for the purpose of describing workflows. They have allowed us to represent every one of the 43 workflow patterns suggested as a catalogue by the Workflow Patterns Initiative.

List of Figures

Bibliography

[1] W. M. P. van der Aalst. Verification of workflow nets. In P. Azema and G. Balbo, editors, *Application and Theory of Petri Nets (LNCS 1248)*, pages 407–426. Springer-Verlag, Berlin, 1997. [cited at p. 27]

[2] W. M. P. van der Aalst. The application of petri nets to workflow management. *Journal of Circuits Systems and Computers*, 8(1):21–66, 1998. [cited at p. 12]

[3] W. M. P. van der Aalst. Workflow verification: Finding control-flow errors using petri-net-based techniques. In W. M. P. van der Aalst, J. Desel, and A. Oberweis, editors, *Proc. BPM'00: Business Process Management (LNCS 1806)*, pages 161–183, 2000. [cited at p. xvi, xviii, xx, 7, 18, 19, 25, 27, 129, 162]

[4] W. M. P. van der Aalst and A. H. M. ter Hofstede. Yawl: yet another workflow language. *Information Systems*, 30(4):245–275, 2005. [cited at p. 12, 19, 132]

[5] W. M. P. van der Aalst, A. H. M. ter Hofstede, B. Kiepuszewski, and A. P. Barros. Workflow patterns. *Distrib. Parallel Databases*, 14(1):5–51, 2003. [cited at p. 21]

[6] W. M. P. van der Aalst and A. H. M. ter Hofstede. Verification of workflow task structures: A petri-net-based approach. *Information Systems*, 25(1):43–69, 2000. [cited at p. 20, 132]

[7] W. M. P. van der Aalst, A. H. M. ter Hofstede, and M. Weske. Business process management: A survey. In W. M. P. van der

Aalst, A. H. M. ter Hofstede, and M. Weske, editors, *Proc. BPM'03: Business Process Management (LNCS 2678)*. Springer-Verlag, Berlin, 2003. [cited at p. 11, 18, 29]

[8] W. M. P. van der Aalst, B. F. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A. J. M. M. Weijters. Workflow mining: A survey of issues and approaches. *Data & Knowledge Engineering*, 47(2):237–267, 2003. [cited at p. 32]

[9] W. M. P. van der Aalst, K.M. van Hee, J.M. van Werf, and M. Verdonk. Auditing 2.0: Using process mining to support tomorrow's auditor. *Computer*, 43(3):90 –93, 2010. [cited at p. 62]

[10] M. C. L. Abeyasinghe, David J. Greenwood, and D. Eric Johansen. An efficient method for scheduling construction projects with resource constraints. *International Journal of Project Management*, 19(1):29–45, 2001. [cited at p. 3]

[11] C. Adam Petri and W. Reisig. Petri net. *Scholarpedia*, 3(4):6477, 2008. [cited at p. 15, 20]

[12] M. Adams, A. H. M. ter Hofstede, D. Edmond, and W. M. P. van der Aalst. Facilitating flexibility and dynamic exception handling in workflows through worklets. In *Short Paper Proceedings of CAiSE'05 Forum*, Porto, Portugal, June 2005. [cited at p. 33]

[13] S. Adolph, P. Bramble, A. Cockburn, and A. Pols. *Patterns for Effective Use Cases*. Addison Wesley, 2003. [cited at p. x, xxi, 35, 39, 43]

[14] M. I. Aguirre-Urreta and G. M. Marakas. Comparing conceptual modeling techniques: a critical review of the EER vs. OO empirical literature. *ACM SIGMIS Database*, 39(2):9–32, 2008. [cited at p. 23]

[15] B. Albert and T. Tullis. *Measuring the user experience*. Morgan Kaufmann, 2008. [cited at p. xii, xiii, xiv, xv, xvi, xix, 23, 44, 45, 46]

[16] C. Alexander, S. Ishikawa, M. Silverstein, M. Jacobson, I. Fiksdahl-King, and S. Angel. *A Pattern Language: Towns, Buildings, Construction*. Oxford University Press USA, 1977. [cited at p. 21, 117]

[17] I. Alexander. Misuse cases: use cases with hostile intent. *Software*, 20, no.1:58–66, Jan/Feb 2003. [cited at p. 35]

[18] I. Alexander and N. Maiden. *Scenarios, stories, use cases: through the systems development life-cycle*. John Wiley and Sons Ltd, England, 2004. [cited at p. 34]

[19] W. S. Ambler. *The Object Primer*. Cambridge University Press, 2004. [cited at p. 13, 36]

[20] H. Anderson, J. Apergis, S.D. Piccolo, C. Geier, C. Kaji, and S. Leisegang. *Professional K2 Blackpearl*. John Wiley & Sons, 2009. [cited at p. 106]

[21] J. E. Bardram. Plans as situated action: an activity theory approach to workflow systems. In *Proc. ECSCW'97: fifth European Conference on Computer-Supported Cooperative Work*, 1997. [cited at p. 33]

[22] J. Billington, S. Christensen, K. van Hee, E. Kindler, O. Kummer, L. Petrucci, R. Post, C. Stehno, and M. Weber. *Applications and Theory of Petri Nets*, chapter The Petri Net Markup Language: Concepts, Technology, and Tools, pages 483–505. 2003. [cited at p. 18, 20]

[23] L. Carrol. *Through the Looking-Glass, and What Alice Found There*. 1872. Available from Project Gutenberg at http://www.gutenberg.org/files/12/12.txt. [cited at p. 115, 133, 145]

[24] A. Cockburn. *Writing Effective Use Cases*. Addison Wesley, 1999. [cited at p. x, xi, xii, xiv, xv, xvi, xvii, xviii, xix, xxi, 35, 36, 38, 39, 40, 41, 42, 43, 56, 64, 67, 90, 129, 211, 212, 224, 228, 234, 262, 263]

[25] Various Contributors. *The Oxford English Dictionary*. 2nd ed, on line edition, June 1989. [cited at p. x, xiii, xiv, xv, xvi, xviii, xix, xx, 3, 6, 25, 41, 50, 67, 72, 73, 83, 97, 120]

[26] K. Cox, A. Aurum, and R. Jeffery. An experiment in inspecting the quality of use case descriptions, 2008. [cited at p. 111]

[27] K. Cox, K. Phalp, and M. Shepperd. Comparing use case writing guidelines. In *Proc. REFSQ'01: Workshop on Requirements Engineering Foundation of Software Quality.* [cited at p. 22, 56]

[28] David Crystal. *How Language Works.* Penguin Group (Australia), 2008. [cited at p. 43]

[29] Evgeny Dantsin, Thomas Eiter, Georg Gottlob, and Andrei Voronkov. Complexity and expressive power of logic programming. *ACM Comput. Surv.*, 33(3):374–425, September 2001. [cited at p. 20]

[30] T. C. Davis, R. F. Holcombe, H. J. Berkel, S. Pramanik, and S. G. Divers. Informed consent for clinical trials: a comparative study of standard versus simplified forms. *Journal of the National Cancer Institute*, 90(9), May 1998. [cited at p. 55, 56, 58]

[31] A.L.L. de Figueiredo, W.L. Andrade, and P.D.L. Machado. Generating interaction test cases for mobile phone systems from use case specifications. *ACM SIGSOFT Software Engineering Notes*, 31, November 2006. [cited at p. 35]

[32] J. Desel and W. Reisig. *Lectures on Petri Nets I: Basic Models*, chapter Place/transition Petri Nets. 1998. [cited at p. 20, 132]

[33] G.. Deutsche. *through the language glass.* Metropolitan Books, Henry Holt and Company, New York, 2010. [cited at p. 1, 50, 51]

[34] A. Dix, L. Finlay, G. Abowd, and R. Beale. *Human Computer Interaction.* Pearson Education, 2004. [cited at p. x, 43, 44, 45, 64, 90, 100, 144, 157]

[35] B. Dushnik and E. W. Miller. Partially ordered sets. *American Journal of Mathematics*, 63(3):600–610. [cited at p. xvi, 39]

[36] Rabbi Eleazar. Babylonic talmud. *Tractate Megillah.* [cited at p. 11]

[37] D. Fahland, C. Favre, B. Jobstmann, J. Koehler, H. Völzer N. Lohmann, and K. Wolf. Instantaneous soundness checking of industrial business process models. In U. Dayal, J. Eder, J. Koehler,

and H. Reijers, editors, *Proc. BPM'09: Business Process Management (LNCS 5701)*, pages 278–293, 2009. [cited at p. 27, 129, 162]

[38] E. B. Fernandez and J. C. Hawkins. Determining role rights from use cases. In *RBAC '97: Proceedings of the second ACM workshop on Role-based access control*, pages 121–125, New York, NY, USA, 1997. ACM. [cited at p. 35, 83]

[39] Diogo R. Ferreira and Daniel Gillblad. Discovering process models from unlabelled event logs. In U. Dayal, J. Eder, J. Koehler, and H. Reijers, editors, *Proc. BPM'09: Business Process Management (LNCS 5701)*, pages 143–158. Springer-Verlag, Berlin, 2009. [cited at p. 32]

[40] M. Fiammante. *Dynamic SOA and BPM*. IBM Press, Pearson PLC, 2010. [cited at p. 57]

[41] M. Fjeld, K. Lauche, M. Bichsel, F. Voorhorst, H. Krueger, and M. Rauterberg. Physical and virtual tools: Activitytheory applied to the design of groupware. *Comput. Supported Coop. Work*, 11(1-2):153–180, 2002. doi = http://dx.doi.org/10.1023/A:1015269228596. [cited at p. 33]

[42] B. Flyvbjerg. Five misunderstandings about case-study research. *Qualitative Inquiry*, 12:219–245, 2006. [cited at p. 92]

[43] Rabbi Jonathan from Bait Govrin. Palestinian talmud. *Tractate Megillah*, pages 4:15a),. [cited at p. 1]

[44] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley Longman, 1995. [cited at p. 21, 117]

[45] A. Gemino and Y. Wand. A framework for empirical evaluation of conceptual modeling techniques. *Requirements Engineering*, 9(4):248–260, November 2004. [cited at p. 22, 146]

[46] A. Gemino and Y. Wand. A framework for empirical evaluation of conceptual modeling techniques. *Requir. Eng.*, 9(4):248–260, 2004. [cited at p. 139, 140, 156]

[47] T. R. G. Green. Cognitive dimensions of notations. In *People and Computers V*, pages 443–460. University Press, 1989. [cited at p. 136, 151]

[48] T.R.G. Green and M. Petre. Usability analysis of visual programming environments: A 'cognitive dimensions' framework. *Journal of Visual Languages & Computing*, 7:131–174(44), June 1996. [cited at p. 24]

[49] V. Gruhn and R. Laue. What business process modelers can learn from programmers. *Science of Computer Programming - Special Issue on Increasing Adequacy and Reliability of EIS*, 65(1):4–13, 2007. [cited at p. 136]

[50] R. Guindon. Designing the design process: exploiting opportunistic thoughts. *Hum.-Comput. Interact.*, 5(2):305–344, 1990. [cited at p. 24]

[51] J.G. Hall, M. Jackson, R.C. Laney, B. Nuseibeh, and L. Rapanotti. Relating software requirements and architectures using problem frames. In *Proc. RE'02: IEEE Joint International Conference on Requirements Engineering*, pages 137–144. IEEE, 2002. [cited at p. 36]

[52] R. Hamadi and B. Benatallah. A petri net-based model for web service composition. In K. Schewe and X. Zhou, editors, *Proceedings of the 14th Australasian Database Conference*, volume 17 of *Conference Proceeding*, pages 191–200, Adelaide, Australia, 2003. ACM International, Australian Computer Society, Darlinghurst, Australia. [cited at p. 20, 132]

[53] P. Heinl, S. Horn, S. Jablonski, J. Neeb, K. Stein, and M. Teschke. A comprehensive approach to flexibility in workflow management systems. In *WACC '99: Proceedings of the international joint conference on Work activities coordination and collaboration*, pages 79–88, New York, NY, USA, 1999. ACM. [cited at p. 62, 111]

[54] J. Hidders, M. Dumas, W. M. P. van der Aalst, A. H. M. ter Hofstede, and J. Verelst. When are two workflows the same? In *Proc. CATS'05: Australasian symposium on Theory of computing*, pages

3–11, Darlinghurst, Australia, Australia, 2005. Australian Computer Society. [cited at p. 120]

[55] J. Hoogland. Change in control. In U. Dayal, J. Eder, J. Koehler, and H. Reijers, editors, *Proc. BPM'09: Business Process Management (LNCS 5701)*, pages 28–30, 2009. [cited at p. 32]

[56] R. Hurlbut. A survey of approaches for describing and formalizing use cases. *Department of Computer Science, Illinois, Institute of Technology*, Technical report:XPT–TR–97–03, 1997. URL http://www.iit.edu/~rhurlbut/xpt-tr-97-03.html. [cited at p. 39]

[57] I. Jacobson. *Object-oriented software engineering.* ACM, New York, NY, USA, 1992. [cited at p. x, 35, 36]

[58] T. Jokela, N. Iivari, J. Matero, and M. Karukka. The standard of user-centered design and the standard definition of usability: analyzing ISO 13407 against ISO 9241-11. In *Proc. CLIHC '03: Latin American conference on Human-computer interaction*, pages 53–60. ACM, 2003. [cited at p. xviii, 44]

[59] Harold Kerzner. *Project Management: A Systems Approach to Planning, Scheduling, and Controlling.* John Wiley & Sons, Inc, 2009. [cited at p. 3]

[60] J. Kim, J. Hahn, and H. Hahn. How do we understand a system with (so) many diagrams? cognitive integration processes in diagrammatic reasoning. *Information Systems Research*, 11(3):284–303, 2000. [cited at p. 23, 24]

[61] G. R. Klare. Assessing readability. *Reading Research Quarterly*, 10(1):62–102. [cited at p. 22, 54]

[62] D. R. Krathwohl. A Revision of Bloom's Taxonomy: An Overview. *Theory Into Practice*, 41(4):212–218, 2002. [cited at p. 148]

[63] K. Kuutti. The concept of activity as a basic unit of analysis for cscw research. In *ECSCW'91: Proceedings of the second conference on European Conference on Computer-Supported Cooperative Work*,

pages 249–264, Norwell, MA, USA, 1991. Kluwer Academic Publishers. [cited at p. 33]

[64] J. H. Larkin and H. A. Simon. Why a diagram is (sometimes) worth ten thousand words. *Cognitive Science*, 11(1):65–100, 1987. [cited at p. xiv, 57, 135, 146]

[65] A. Lauder and S. Kent. Precise visual specification of design patterns. In E. Jul, editor, *Proc. ECOOP'98: 12th European Conference on Object-Oriented Programming*, pages 114–134, 1998. [cited at p. 21]

[66] W. J. Lee, S. D. Cha, and Y. R. Kwon. Integration and analysis of use cases using modular petri nets in requirements engineering. *IEEE Transactions on Software Engineering*, 24(12):1115–1130, 1998. [cited at p. 20, 57, 121, 129, 162]

[67] D. Leffingwell and D. Widrig. *Managing software requirements: a unified approach*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2000. ISBN 0-201-61593-2. [cited at p. 35, 64]

[68] N. A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996. ISBN 1-55860-348-4. [cited at p. 121]

[69] G. Reid Lyon, Sally E. Shaywitz, and Bennett A. Shaywitz. A definition of dyslexia. *Annals of Dyslexia*, 53(1):1–14, 1 2003. [cited at p. 53]

[70] E. W. Mayr. An algorithm for the general petri net reachability problem. In *STOC '81: Proceedings of the thirteenth annual ACM symposium on Theory of computing*, pages 238–246, New York, NY, USA, 1981. ACM. [cited at p. 20]

[71] G. Harry McLaughlin. Smog grading — a new readability formula. pages 639–646, May 1969. [cited at p. xvii, 54, 55, 58]

[72] J. Mendling, H. A. Reijers, and J. Cardoso. What makes process models understandable? In G. Alonso, P. Dadam, and M. Rosenmann, editors, *Proc. BPM'07: Business Process Management (LNCS 4714)*, pages 48–63, 2007. [cited at p. 24, 145]

[73] S. Milgram. *Obedience to Authority: An Experimental View*. Harper and Row, New York, 1974. [cited at p. x]

[74] H. Mili, G. Tremblay, Guitta Bou Jaoude, É. Lefebvre, L. Elabed, and Ghizlane El Boussaidi. Business process modeling languages: Sorting through the alphabet soup. *ACM Comput. Surv.*, 43(1):4:1–4:56, December 2010. [cited at p. 12, 13, 17]

[75] George A. Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *The Psychological Review*, 63:81–97, 1956. [cited at p. 44]

[76] T. G. Moher, D. C. Mak, B. Blumenthal, and L. M. Leventhal. Comparing the comprehensibility of textual and graphical programs: the case of petri nets. In *Empirical Studies of Programmers: Fifth Workshop*, 1993. [cited at p. 57, 135]

[77] D. Moody. The "physics" of notations: Toward a scientific basis for constructing visual notations in software engineering. *IEEE Transactions on Software Engineering*, 35:756–779, November 2009. [cited at p. 64, 146]

[78] Peter W. G. Moris. *The management of projects*. Thomas Telford Services, 1994. [cited at p. 4]

[79] T. Murata. Petri nets: Properties, analysis and applications. volume Vol. 77, no. 4 of *Proceedings of the IEEE.*, pages 541–580, 1989. [cited at p. 12, 13, 15, 20]

[80] B. A. Nardi, editor. *Context and consciousness: activity theory and human-computer interaction*. Massachusetts Institute of Technology, Cambridge, MA, USA, 1995. [cited at p. 33]

[81] J. Nielsen. *Usability Engineering*. Academic Press, Inc., San Diego, 1993. [cited at p. 21, 44]

[82] Jakob Nielsen and Thomas K. Landauer. A mathematical model of the finding of usability problems. In *Proc. CHI '93: ACM conference*

on Human Factors in Computing Systems, pages 206–213, New York, NY, USA, 1993. ACM. [cited at p. 144]

[83] D. Norman. Cognitive engineering. In *Norman, D. and Draper, S. Editors; User centered design: new perspectives on human computer interaction*, pages 31–61. Hillsdale, 1986. [cited at p. xvii, 22, 23]

[84] M. Notomi and T. Murata. Hierarchical reachability graph of bounded petri nets for concurrent-software analysis. *IEEE Transactions on Software Engineering*, 20(5):325–336, May 1994. [cited at p. xvii, 20, 124]

[85] Office of Government Commerce of Great Britain. *Managing successful projects with PRINCE2*. The Office of Government and Commerce under licence from her Majesty's Stationary Office. [cited at p. xvii, 4]

[86] A. Ottensooser and A. Fekete. An enactment-engine based on use-cases. In G. Alonso, P. Dadam, and M. Rosemann, editors, *Proc. BPM'07: Business Process Management (LNCS 4714)*, pages 230–245, Heidelberg, 2007. Springer-Verlag. [cited at p. vii, 8, 34, 57, 78, 79]

[87] A. Ottensooser and A. Fekete. Workflow patterns represented in use-cases. Technical report, School of Information Technologies, University of Sydney, Australia, 2008. TR-611, ISBN 978-1-74210-021-0. [cited at p. viii]

[88] A. Ottensooser and A. Fekete. Comparing readability of graphical and sentential process design notations – data analysis report. Technical report, School of Information Technologies, University of Sydney, Australia, 2010. TR-658, ISBN 978-1-74210-198-9. [cited at p. vii, 143, 145, 150, 153]

[89] A. Reijers A.H. Mendling j. Ottensooser, A. Fekete and C. Menictas. Making sense of business process descriptions: An experimental comparison of graphical and textual notations. *J. Syst. Software*, to appear 2011. [cited at p. vii]

[90] S. Paracchini, C.D. Steer, L.L. Buckingham, A.P. Morris, S. Ring, and et al. Association of the kiaa0319 dyslexia susceptibility gene

with reading skills in the general population. *J Psychiatry*, 2008. [cited at p. 53]

[91] J. Parsons and L. Cole. What do the pictures mean? guidelines for experimental evaluation of representation fidelity in diagrammatical conceptual modeling techniques. *Data & Knowledge Engineering*, 55(3):327342, 2005. [cited at p. 139, 140, 156]

[92] S. Pinker. *The Language Instinct*. Penguin Book, Kindle Edition, London, 1994. [cited at p. 47, 48, 49]

[93] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2005. [cited at p. x, 149]

[94] H.A. Reijers and J. Mendling. A study into the factors that influence the understandability of business process models. *IEEE Transactions on Systems, Man, and Cybernetics – Part A*, 2010. [cited at p. 139, 140, 156]

[95] N. Russell, W. M. P. van der Aalst, A. H. M. ter Hofstede, and D. Edmond. Workflow resource patterns: Identification, representation and tool support. In O. Pastor and J. Falcao e Cunha, editors, *Proc. CAiSE'05: International Conference on Advanced Information Systems Engineering (LNCS 3520)*, pages 216–232. Springer-Verlag, 2005. [cited at p. xx, 29]

[96] N. Russell, A. H. M. ter Hofstede, W. M. P. van der Aalst, and N. Mulyar. Workflow control-flow patterns: A revised view. Technical Report BPM Center Report BPM-06-22, BPM Center, 2006. [cited at p. 12, 21, 66, 70, 117, 124, 127, 131]

[97] N. Russell, P. Wohed, W. M. P. van der Aalst, M. Dumas, and A. H. M. ter Hofstede. chapter Pattern-Based Analysis of the Control-Flow Perspective of UML Activity Diagrams, pages 63–78. 2005. [cited at p. 13, 15, 21, 116, 120]

[98] S. S. Shapiro and M. B. Wilk. An analysis of variance test for normality (complete samples). *Biometrika*, 52(3-4):591611, 1965. [cited at p. 152]

[99] S. Shaywitz. *Overcoming Dyslexia: A New and Complete Science-Based Program for Reading Problems at Any Level.* 2003], publisher = Alfred A. Knopf. [cited at p. 51, 52, 146]

[100] L. Si and J. Callan. A statistical model for scientific readability. In *Proc. CIKM'01: tenth international Conference on Information and Knowledge Management*, pages 574–576, New York, NY, USA, 2001. ACM. [cited at p. 22, 58]

[101] Luo Si and Jamie Callan. A statistical model for scientific readability. In *Proceedings of the tenth international conference on Information and knowledge management*, CIKM '01, pages 574–576, New York, NY, USA, 2001. ACM. [cited at p. 54, 55]

[102] K. Siau. Informational and computational equivalence in comparing information modeling methods. *Journal of Database Management*, 15(1):73–86, Jan.-Mar 2004. [cited at p. 146]

[103] K. Siau and M. Rossi. Evaluation techniques for systems analysis and design modelling methods – a review and comparative analysis. *Information Systems Journal*, 2008. [cited at p. 22]

[104] StataCorp Corporation. *Stata – Statistical Software: Release 7.0.* College Station, TX, 2001. [cited at p. 149]

[105] G. Steiner. *After Babel: Aspects of Language and Translation.* Oxford Univeristy Press, Oxford, 1998. [cited at p. 47, 50]

[106] R. N. Taylor and A. van der Hoek. Software design and architecture: The once and future focus of software engineering. In *FOSE '07: Future of Software Engineering*, pages 226–243, Washington, DC, USA, 2007. IEEE Computer Society. [cited at p. 21]

[107] S. Taylor. Extreme terseness: Some languages are more agile than others. In M. Marchesi and G. Succi, editors, *Proc. XP'03*, pages 334–336, 2003. [cited at p. 21]

[108] F. Tørner, M. Ivarsson, F. Pettersson, and P. Öhman. Defects in automotive use cases. In *ISESE '06: Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering*, pages 115–123, New York, NY, USA, 2006. ACM. [cited at p. 56]

[109] K. van Hee, O. Oanea, R. Post, L. Somers, and J. M. an der Werf. Yasper: a tool for workflow modeling and analysis. In *Proceedings of the Sixth international Conference on Application of Concurrency To System Design*, pages 279–282, Washington, DC, 2006. ACSD. IEEE Computer Society. [cited at p. x]

[110] I. Vessey. Cognitive fit: A theory-based analysis of the graphs versus tables literature. *Decision Sciences*, 22(2):219–240, 1991. [cited at p. 137]

[111] C. Ware, H. Purchase, L. Colpoys, and M. McGill. Cognitive measurements of graph aesthetics. *Information Visualization*, 1(2):103–110, 2002. [cited at p. 21, 24]

[112] S. A. White. Introduction to BPMN. *BPTrends*, July 2004. [cited at p. xiii, 13, 14]

[113] F Wilcoxon. Individual comparisons by ranking methods. *Biometrika*, 1(6):80–83, December 1945. [cited at p. 152]

[114] James M. Wilson. Gantt charts: A centenary appreciation. *European Journal of Operational Research*, 149(2):430 – 437, 2003. Sequencing and Scheduling. [cited at p. 2]

[115] P. Wohed, W. M. P. van der Aalst, M. Dumas, A. H. M. ter Hofstede, and N. Russell. On the Suitability of BPMN for Business Process Modelling. In S. Dustdar, J. Fiadero, and A. Sheth, editors, *Proc. BPM'06: Business Process Management (LNCS 4102)*, pages 161–176, 2006. [cited at p. 15, 21, 66, 116, 120]

[116] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén. *Experimentation in software engineering: an introduction*. Kluwer Academic Publishers, Norwell, MA, USA, 2000. [cited at p. x, 25, 134]

[117] Workflow Management Coalition. Terminology and glossary. Technical report, February 1999. WFMC-TC-1011. [cited at p. xii, xiii, xvii, xx, xxi, 12, 29, 30, 31, 33, 35]

[118] Edward Yourdon and Larry L. Constantine. *Structured design : fundamentals of a discipline of computer program and systems design*. Yourdon Press, 2nd edition, 1979. [cited at p. 36]

[119] P. Zave. Classification of research efforts in requirements engineering. In *Proceedings of the Second IEEE International Symposium on Requirements Engineering*, pages 214 – 216, mar 1995. [cited at p. 34]

Index