Declarative Service Modeling through Adaptive Case Management

Evan Morrison

School of Computer Science and Software Engineering
University of Wollongong

Abstract. Adaptive Case Management addresses the shift away from the prescriptive process centric view of operations towards a declarative framework for operational descriptions that promotes dynamic task selection in knowledge intensive operations. A key difference between prescriptive services and declarative services is the way by which control flow is defined. Repeatable and straight-thru processes have been successfully used to model and optimise simple activity based value chains. Increasingly, traditional process modelling techniques are being applied to knowledge intensive activities with often poor outcomes. By taking an adaptive case management approach to knowledge intensive services, it is possible to model and execute workflows such as medical protocols that have previously been too difficult to describe with typical BPM frameworks. In this article we describe an approach to design level adaptive case management leveraging off existing repositories semantically annotated business process models.

Keywords: Adaptive Case Management, BPM, Requirements Engineering, Governance, Effects, Process Modeling

1 Introduction

Research in business process management (BPM) has recently been heavily focused on the operational activities of organisations. One crucial research challenge in the space is that of activity fulfilment in dynamic environments. Recently, there has been an increasing drive for the examination of case handling and management as an alternative to traditional straight-thru processing (Zhu et al., 2013). Straight-thru processing deals with the construction and operation of repeatable workflow and process designs, whereas, case management investigates roles, life cycle, and activity implementation from a more consumer or interactional point of view (Ly, Rinderle-Ma, Göser, & Dadam, 2012). This chapter addresses the shift away from the prescriptive process centric view of operations toward a declarative framework for operational descriptions. This movement promotes intelligent task diversity in knowledge intensive operations. A key difference between prescriptive processes and declarative process is the definition of control flow. Repeatable and straight-thru processes allow easy modeling and optimisation of basic activity based value chains (Morrison, Ghose, Dam, Hinge, & Hoesch-Klohe, 2012). Poor outcomes typically result from the application of process modeling techniques to knowledge intensive activities (Zhu et al., 2013). An adaptive case management (ACM) approach to process management makes it possible to create knowledge intensive workflows that are not possible to model using traditional BPM systems.

"Case management is built around the concept of processing a case, a collection of information and coordinated activities, by organisational knowledge workers"(Zhu et al., 2013). Typically, a case is a focused view of an interaction with a business unit or organisation

by an external entity (customer). A customer, driven through some desire or need, engages with an organisation. These engagements typically result in mutual exchange for services and resources. Through these interactions, various processes compositions and choreographies appear to create a semi coherent procedure aimed at satisfying the customer's primary goals or desires. This differs from traditional workflows, which make personalised customer transactions and narrative based progressions impossible. Prescriptive processes also typically mean that a customer engaging at multiple touch points will need to repeat activities, such as explaining goals several times for each process context. Within a case management framework, a customer engagement case contains all details of the customer's goals and past interactions, sharing case details throughout the execution cycle. As such, all relevant data and information are available from within the processes and also for the composition engine.

During the creation of workflow systems, process designers strive to create process models and designs that benefit many varying use cases (Bleistein, Cox, & Verner, 2006; Edirisuriya & Johannesson, 2009; Wang & Ghose, 2010). The problem for these activities is in defining processes that can be used in varying contexts based on customer demands and intentions (Koliadis, Ghose, & Padmanabhuni, 2008; Wang & Ghose, 2010; Gordijn, Soetendal, & Paalvast, 2005). Being able to dynamically construct a process that effectively works to satisfy consumer goals as well as business goals taking into consideration operational and historic context is necessary in a business setting (Koliadis & Ghose, 2006; Koliadis et al., 2008; Gordijn et al., 2005). To ensure that enterprise context is realised we presuppose that process models contribute to some part of an organizational strategy and also satisfy customer desires.

In this chapter, we describe the use of adaptive case management (Khomyakov & Bider, 2000; Mundbrod, Kolb, & Reichert, 2012; Strijbosch, 2011; Hildebrandt, Mukkamala, & Slaats, 2011b, 2011a) and strategic alignment (Morrison et al., 2012) to model task composition and choreography that will provides declarative support to organizations with existing process management solutions. This work leverages research into semantically annotated tasks for meaningful business process modeling.

Using the declarative case management framework presented in this chapter, organization will be able to leverage their existing process management systems to begin to transition and support new knowledge intensive services in an adaptive case management context.

This chapter is broken into the following order. In §2, we have provided an example scenario that describes a generic complaint handling process, this process is used to demonstrate a scenario where declarative service modeling is beneficial and is also used throughout the chapter to provide details on key concepts. In §3, we provide a background of the language that form the basis of this framework. In §4 we provide the key chapter contributions. First, we describe a mechanism for computing end-to-end scenarios using existing organisational process definitions. Secondly, we show how to compute the semantic state of new constructed case management sequence flows. We compare our work to existing literature in §7, then conclude and position our future work in §8.

2 Motivating Example

Throughout this chapter, we will use a fictionalised motivating example for telephone company *VirgaFone* to introduce the conceptual building blocks and finally a framework of

dynamic and declarative service composition. The knowledge systems that we work with are shown in the overview of VirgaFone in figure 1. These are the strategic landscape, process repository, business rules and knowledge base. The strategic landscape includes details of the rationale and desires that VirgaFone wishes to achieve with it's call centre. A process repository is used to maintain all operational activities that VirgaFone is capable of performing. Business rules are integrated into processes to maintain application logic across the operational elements of VirgaFone. A knowledge base of key definition and logical correlations is used to support decision making between strategic and operational levels.

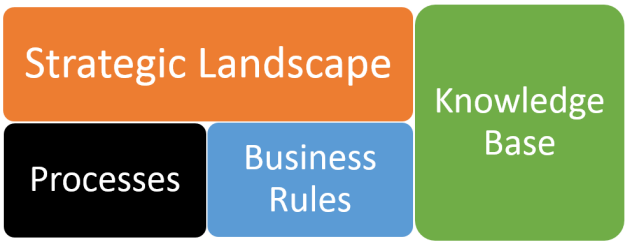

Fig. 1. VirgaFone Call Centre Overview

Process Repository The example case for the company *VirgaFone* described here is of a *complaint handling process*. The complaint handling process is a commonly servitised process that is deployed across in most telecom customer call centres. The process has been depicted in figure 2 using the business process modeling notation (the notation is discussed in §3).

The process flows as follows, if a customer is upset at any point in time with the level of service or products that they have received through a transaction with *VirgaFone*, they will call the call centre and make a complaint. The first stage of a complaint handling process is to ensure that the customer details are recorded, and as such verifying customer details is the first task completed by the call centre agent. In the event that a customer can not be verified over the phone then they will be directed to make their complaint in person at the closest local store or business kiosk. This is marked in the model as an orange task, as it is a *non-intentional* or *exceptional* task (discussed in §3).

Business Rules After a customer has been verified, then the complaint is recorded and assessed against a collection of business rules. Business rules can be represented in any number of formations, including through SBVR (Semantics of Business Vocabulary and Business Rules), plain text and processable rules formats (for example drools rules language). To avoid confusion we are using a Drools spreadsheet encoding of a set of simple decision rules. These are shown in figure 3, the first section describes the rule namespace and reference classes, the second section describes a set of mapping from state to conditions for the process flow. Currently, the rules state that if a complain is made about long waits, that the process should be directed to the close case task, otherwise if the customer is experiencing call drop outs that their case should be escalated for further customer satisfaction management.

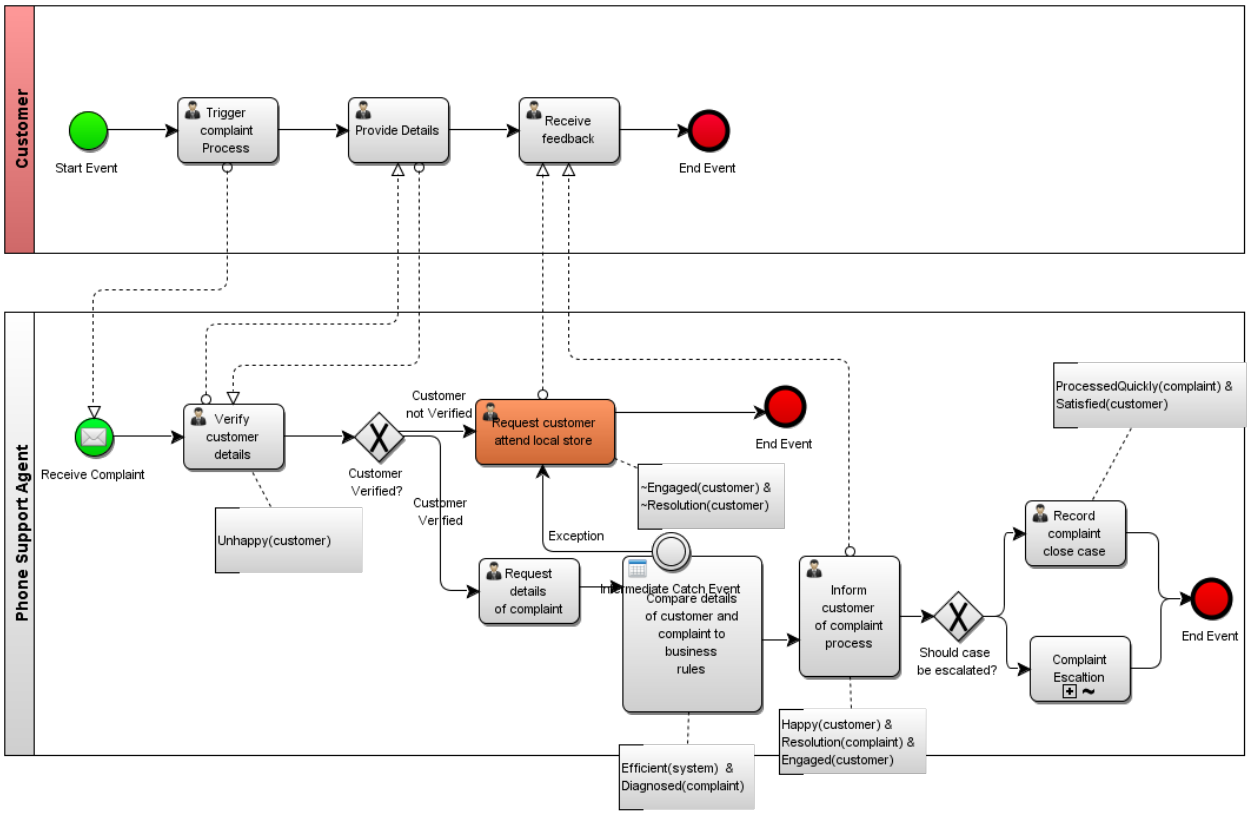

Fig. 2. VirgaFone Complaints handling process

	RuleSet Import Notes	au.edu.uow.dsl au.edu.uow.dsl.DecisionTable.Message This decision table is used for complaint handling escalations	
	RuleTable VirgaFone		
	CONDITION	ACTION	ACTION
	m:Message		
	status == $param	m.setMessage("$param");update(m);	m.setStatus($param);update(m);
Complaint Handling Rules	**Status**	**Set message**	**Set status**
Call Drops out lots	**Message.DROPOUTS**	**Service level complaint**	**Message.ESCALATE**
Long wait time for complaints	**Message.LONGWAIT**	**Low Level Complaint**	**Message.CLOSE**
Long wait time for complaints	**Message.LONGWAIT_AND_DROPOUTS**	**Service level complaint**	**Message.ESCALATE**
No complaint made	**Message.NOCOMPLAINT**	**No Complaint Recorded**	**Message.CLOSE**
Escalate	**Message.ESCALATE**		
Close Case	**Message.CLOSE**		

Fig. 3. Decision rules for VirgaFone complaints handling process

When the process has been completed, either the customers will have been directed to a business centre / kiosk, the customers complaint will have been recorded or the customers complaint will have been escalated for further processing by a customer engagement officer.

Strategic Landscape A strategic landscape is a list of strategic goals that describe core values that the call centre wants to realize. We have used an i^* styled goal notation to model strategies. A goal model, describes the organisational desires and intentions, they guide specific process deployments with purpose and reason. A goal model describes the *why* of operations. Notational hard goals (rounded ellipses) are used to describe functional goals, and soft goals (cloud shaped) are used to describe optimisation objectives. In §3, these concepts are discussed in finer detail. Because this process will be run in a dynamic environment, where call load and exceptional scenarios highly possible to occur, a goal model of VirgaFone's aims and objectives can be used to inform decision makers when process improvement activities are performed. In figure 2, a goal model of the call centre has been provided and describes the heirarchy of soft / hard goals for the call centre.

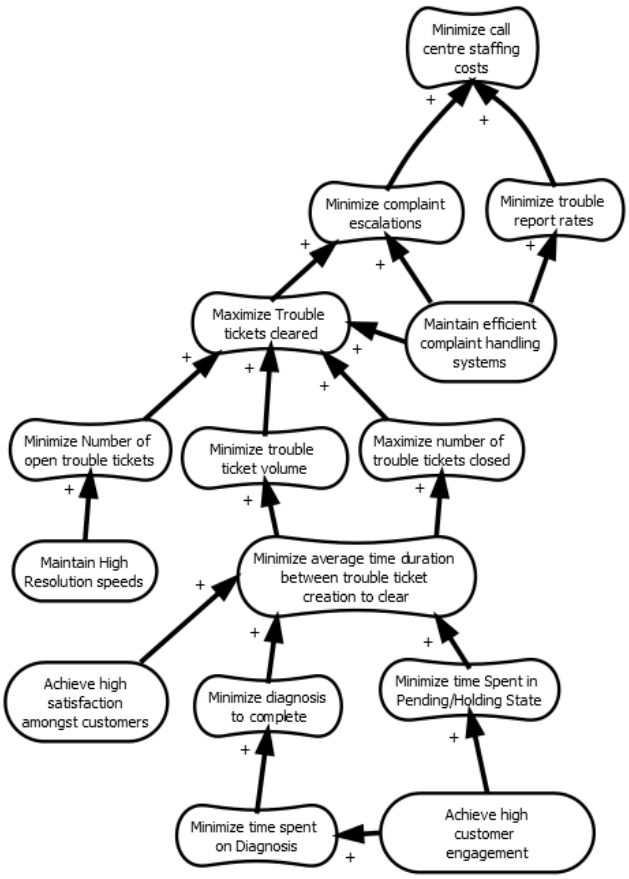

Fig. 4. Goal model for VirgaFone Call Centre

Each functional goal usually has an included predicate description that can help during strategic alignment. The strategies of the VirgaFone call centre are as follows:

- (Optimisation) Minimize call centre staffing costs
- (Optimisation) Minimize complaint escalations
- (Optimisation) Minimize trouble report rates

- (Optimisation) Maximize Trouble tickets cleared
- (Optimisation) Minimize Number of open trouble tickets
- (Optimisation) Minimize trouble ticket volume
- (Optimisation) Maximize number of trouble tickets closed
- (Optimisation) Minimize average time duration between trouble ticket creation to clear
- (Optimisation) Minimize diagnosis to complete
- (Optimisation) Minimize time Spent in Pending/Holding State
- (Optimisation) Minimize time spent on Diagnosis

- (Goal) Maintain efficient complaint handling systems
- (Goal) Maintain High Resolution speeds
- (Goal) Achieve high satisfaction amongst customers
- (Goal) Achieve high customer engagement

Knowledge Base Artefacts from different levels of abstraction generally are described in different languages and with differing levels of coarseness, it is essential to maintain a data dictionary and knowledge base that describes key concepts and can be used to translate between the concepts. A domain specific knowledge base that describes the call centre is provided below as a set of rules are written as a knowledge dictionary i.e. $A \Rightarrow B$. Whenever A is observable then B exists.

- $\mathit{Resolution(complaint)} \wedge \mathit{(Execution(time)} \leq \mathit{10min)} \Rightarrow \mathit{Resolution(complaint,high\text{-}speed)}$
- $\frac{|\mathit{instance(Resolution(complaint,high\text{-}speed))}|}{|\mathit{instance(all)}|} \geq .55 \Rightarrow \mathit{Maintain\text{-}High\text{-}Resolution\text{-}speeds}$
- $\frac{|\mathit{instance(Resolution(complaint))}|}{|\mathit{instance(all)}|} \geq .75 \Rightarrow \mathit{Efficient(ComplaintSystem)}$
- $\mathit{Efficient(ComplaintSystem)} \Rightarrow \mathit{Maintain\text{-}efficient\text{-}complaint\text{-}handling\text{-}systems}$
- $\mathit{Processed(case)} \wedge \mathit{Satisfied(customer)} \Rightarrow \mathit{Achieve\text{-}high\text{-}satisfaction\text{-}amongst\text{-}customers}$
- $\frac{|\mathit{instance(Engaged(customer))}|}{|\mathit{instance(all)}|} \geq .75 \Rightarrow \mathit{Achieve\text{-}high\text{-}customer\text{-}engagement}$

The knowledge based acts as a translator between instance level, design level and strategic level, and as such the languages used will appear non-standard, for ease of understanding **instance level knowledge is written in green and is denoted using controlled natural language propositions**, **design level knowledge is written in blue and is denoted using controlled natural language propositions**, finally **strategy level knowledge is written as classic propositions and are marked in orange**. Defining and using such a knowledge base is an enabler for businesses. Using this knowledgebase with a framework such as that described in (Morrison et al., 2012) will allow an organisation to conduct strategic analysis, to determine if all of it's processes contribute towards it's strategic objectives.

Scenario VirgaFone has conducted a detailed business analysis project to obtain all of the above BPM artefacts, and now seeks to leverage the information to become more adaptive in their dynamic operating environment. Call centre managers have observed that in the event that there is a large network outage in a particular area, their customers begin to complain heavily. When this happens, VirgaFone's normal customers have high wait times and become disgruntled with a high churn rate (a customer churns from a network by switching carriers).

VirgaFone would like to lower these churn occurrences and has invested heavily in a scalable call centre (during peak demands, the call centre requests resources from external call centres or other call departments); however, the additional call centre team are not as flexible or knowledgeable about the complaints process as existing staff members. Though has now found that by simply throwing more bodies at it's problem that it's costs are increasing beyond the losses made through the churn process.

To resolve this issue VirgaFone has begun to implement a dynamic and declarative service system using adaptive case management. In this case, as the call staff are onboarded, the process definitions and business rules are dynamically reconfigured to support a smaller number of underskilled operators, while maintaining optimal churn losses to reduce overall losses during network outages.

3 Background

In this section, we will introduce the set of languages used to describe process models and strategies. In the follow sections, we will use these languages to produce a succinct description of strategic business process alignment.

3.1 Business Process Models & Semantic Effects

A process model can be viewed as a representation of a set of activities, and decision rules. The purpose of process modeling is to show the activities that actors or systems perform. Each process should represent a specific, repeatable set of steps. Generally process models are created from a combination of activities, decisions and sequence flow. An activity is a single unit of operation and describes a specific step that can be completed. A decision is a point where the model splits. Depending on environmental state or a user choice, only one path is generally chosen to follow during the execution of a process. Sequence flow is the transition between two activities. Sequence gives order to the activities and decisions in a process. Processes are created using activities representing operations at the same level of granularity, i.e. the statement 'verify customer details' and the statement 'ask the user for their first name' should not appear in the same process. Each process may itself become an activity in a more abstract process model, i.e. the activity 'ask the user for their first name' may occur in a process 'verify customer details'. The Business Process Modeling Notation, BPMN (Dijkman & Van Gorp, 2011), is a standardized notation for creating process models, used to represent business specific process models. It is formed using a collection of activities, gateways, events, sequence flows, pools, swim lanes, and message flows. In this work, whenever a process model is referred to, we mean it to have the qualitities and elements of a BPMN process model. A set of business process models is refered to as a process portfolio. In much the same way as an investment portfolio, a process portfolio is representative of an organizations process assets based on their existing needs and functional requirements.

In a process model, gateways can be used to describe choices and other types of process decisions. XOR gateways, can be used to represent exclusive choice and branches the execution of a process model depending on either user choice, or environmental variables. AND gateways are used to represent a parallel sequence splitting. When a process model splits via an AND gateway, activities of all outgoing branches of the gateway are run independent of

the other branches. The sequence between activities on the same branch is still maintained. There are other types of gateways, the reader is referred to (Dijkman & Van Gorp, 2011) for complete executional semantics.

An event in a process model, is a triggering object, and can be used to instantiate or cease execution through the process, they can also be used to handle intermediate events, such as timer alarms when a process is taking too long. Typically all process models have both a start and an end event. A start event is used as the execution entry point in the process model. An end event is used as the execution termination point of the process model. Once an execution sequence reaches an end event then the instance of the process model has come to an end. It is possible to have more than one starting event and more than one end event in a given model, though for more than one start event the process needs to clearly define a default start point.

Sequence flow in a process model generally goes from one starting event through a series of activities and gateways to the end event. However it is also very possible to have multiple start events and multiple and end events, signifying different triggering or entry points in the model and different ending points in the model.

An effect scenario ϵ describes the result of executing an instance of an activity or part of a process model. The effect scenarios of activities in a process model are given with semantic annotations that describe the resulting changes of state brought about by executing the activity. In this paper we represent each effect as a prime implicant proposition and consider a set of effects as a sentence constructed by the conjunction of the propositions in the set. Through a mechanism of accumulation of effect scenarios, the effects of a process model can be found. Hinge (Hinge, Ghose, & Koliadis, 2009), has described a function for accumulation that takes two effect scenarios and returns a consistent effect scenario that is the accumulated effect scenario. This accumulation can be done in a pairwise manner across a process model to find the effect scenarios of the process.

In a similar fashion, Hoesch-Klohe's (Hoesch-Klohe, Ghose, & Le., 2010) framework Abnoba, can be used to annotate QoS capabilities υ to activities in a process model. These activities can then be pairwise accumulated, to show capabilities for the process.

There are various encodings of process models. Typically, the encodings either find a basis in the space of Petri-net's or graphs. The most common encoding for process models is a WF-net (workflow network) described by van der Aalst (Van Der Aalst, 1998), however, is not suitable for this work because of the nature of execution traces. For this article, a directed graph process model has been used; however, the authors contend that the definitions of alignment will fit the process encodings definitions of Dijkman et. al. (Dijkman & Van Gorp, 2011). Dijkman et. al. have described an extensive BPMN graph encoding including the execution semantic of a large selection of BPMN elements, which is by far too detailed to describe in the space allowed. A process model as a graph is a strongly connected directed graph, where activities events and gateways are nodes in the graph and sequence flow are the edges. Each node in the graph is labelled with a type, a name, and extra attributes such as QoS values and semantic effects.

3.2 Strategy Modeling Language

When thinking at an organizational level, it is common to focus on the notion of a strategic plan. Generally, a strategic plan can be broken down into the following headings[1]:

- Organizational description that includes, its structure, vision and mission statement.
- Context, describing the industry scope as trends, advances, opportunities, and weaknesses. The organizational context will also include information about governance and distinctive competencies that make the organization unique.
- Risks including both new and existing. The strategy should list ways that the organization has identified to mitigate these risks.
- A strategy model is a hierarchical definition of strategies that the organization believes will help move it forward while also addressing contextual issues and identified risks. Each strategy usually has an aim, purpose, constraints and stakeholders.
- Processes / Programs are lists of the operational mechanisms that an organization will use to realize strategies in the strategy model.

In previous work (Ghose, Lê, K, & Morrison, 2010), we have proposed a language that can be used by senior executives for describing organizational strategies as part of the strategy model in a strategic plan. This language is the strategy modeling language (SML). The core modeling elements of SML are: Functional Goals, Plans, and Optimization Objectives. These elements can be used in combination to describe a typical strategy model.

There has been a vast amount of work describing strategy modeling such as that done in (Ghose et al., 2010). A strategy model in the strategy modeling language (SML)(Ghose et al., 2010) can be constructed to describe an organization's strategies. SML was developed to provide analysts with a crisp language for describing organization strategies using *Goals, and Plans*, *Optimization Objectives*. Using SML we have established an alignment function that maps cumulative effects to strategic goals.

For the most part, goals are the building block of strategy description languages. When describing a goal, various requirements are encoded (as part of the goal description) to the goal G. When the goal's requirements are achieved the goal is realized. For an introduction to goals and methodologies that exist for goal interpretation the reader is referred to (van Lamsweerde, 2001). Functional Goals describe outcomes that organizations would like to achieve. When written in SML, these can be evaluated as either fulfilled or not fulfilled. Functional goals are used to reflect internal and external realities that an organization wishes to achieve, and generally address strengths, weaknesses and opportunities that have been identified in a SWOT analysis.

In most strategic plans, it is common to identify functional goals that for the short, medium and long terms. When these goals are explicitly ordered then they become part of a strategic plan. Each plan in SML describes milestones in an organizational strategy. A plan is an ordered sequence of functional goals. Plans may follow tactical decisions that describe a means to realize higher-level goals.

An optimization objective in SML is used to discriminate over preferences for strategic outcomes. An optimization objective is typically either the maximize or minimize of a function on a set of given QoS capabilities.

[1] Our current work on the breakdown of the structure of an organizational strategy can be found at `http://www.dsl.uow.edu.au/~edm92/strategy/ont/`

A strategy S is either a plan L or a functional goal G or an optimization objective O. A strategy model $\mathcal{S}$ is a set of all strategies that are to be analysed.

4 Task Composition and Choreography

Recall our example scenario §2, where *VirgaFone* wishes to recompute their operational stack to meet the demands of their changing environment. In the first instance, it would be ideal to be able to recompute the process model that they follow, to better manage their workforce. To recompute a process model, it's graph encoding needs to be mathematically analysed. In the following section we provide a method for structuring process models in graph form and then algorithmically reconstruct the model into new compositions and choreography's to meet the requirements of a dynamic end-to-end customer engagement.

A process model as a graph is a strongly connected directed graph, where activities events and gateways are nodes in the graph and sequence flow are the edges. Each node in the graph is labelled with a type, a name, and extra attributes such as QoS values and semantic effects.

Definition 1: *Semantically Annotated Process Model*
A process model is a labeled directed graph $p = \langle N, F, l, \Omega, \psi, \varphi \rangle$ *with the following properties:*

1. N *is a finite non-empty set of nodes.* ψ, φ *are the start and end nodes respectively and* $\psi, \varphi \in N$.
2. F *a set of control flow links,* $F \subseteq N \times N$.
3. Ω *is a set of labels, each label* ω *is of the form* $\langle$*type, name,* $\langle \upsilon, \epsilon, \gamma \rangle\rangle$. *Where* type *is the type of element, i.e. event, activity, gateway, task.* name *is the name of the element.* $\langle \upsilon, \epsilon, \gamma \rangle$ *is a tuple representing the QoS capabilities, effect scenario and customer state of the node respectively.*
4. $l : N \rightarrow \Omega$ *is a labeling function that assigns labels to nodes of the process model.*
5. $\forall n \in N, (n, \psi) \notin F \wedge (\varphi, n) \notin F$ *i.e., the start node has no incoming edges and the end node has no outgoing edges.*

□

From our example, in figure 2, each activity, event, pool, and gateway will be encoded as a node in a graph. Sequence and message arrows will be encoded as edges connecting two nodes, and annotations such as *'efficient(system)'* will be encoded as the effect scenario ϵ for the labeled node *'compare details of customer and complaint to business rules'*.

A process model $p = \langle N, F, l, \Omega, \psi, \varphi \rangle$ is well formed if a strongly connected graph can be formed from its nodes and edges, i.e., $(N, F \cup (\varphi, \psi))$ is a strongly connected graph. A well formed process model p is a well formed decision free process if there are no XOR gateways in the process model. There are procedures that can be used to construct a collection of well formed decision free process models from a given well formed process model[2]. In this work, we assume that all processes are well formed decision free process models. We denote a process portfolio of well formed decision free process models $\mathcal{P}$.

An example of a decision free version of the complaint handling process is $\langle$ {*[Receive complaint], [verify customer details], [customer verified?], [request customer attend local store],*

[2] A supporting toolkit of libraries implementing most functions described here can be downloaded from `http://www.dsl.uow.edu.au/~edm92/textseer/`, including a procedure to create decision free process models

[end event]$\}, \{\langle$*[Receive complaint], [verify customer details]* $\rangle, \langle$ *[verify customer details, customer verified?]*$\rangle, \langle$ *[customer verified?, request customer attend local store]*$\rangle, \langle$ *[request customer attend local store], [end event]*$\rangle\}, \Omega$,*[Receive complaint], [end event]*$\rangle$, where Ω is the set of labels and associated effects, i.e. $\{\langle$ *Activity, [Verify customer details]*,$\langle \emptyset,$ *unhappy(customer)* $, \emptyset\rangle\rangle, \langle$ *Activity, [Request customer attend local store]*,$\langle \emptyset, \neg$*engaged(customer)* $\wedge \neg$*resolution(customer)* $, \emptyset\rangle\rangle\}$.

The result of accumulating effect scenarios and QoS capabilities in process models is a set of effect scenarios and QoS capabilities that describe the entire process model. There are occasions when it is beneficial to find the effect scenarios or QoS capabilities of a particular instance (or trace) of a process model. A trace is a sequence of activities showing a possible execution instance of the given process. Each trace begins at the start of the process model and continues along to activities in the process until a given point within the process model. To find these traces, either sequential paths, parallel paths, or a combination of the two must be used to describe the instance of the process model of interest.

Given a process model $p = \langle N, F, l, \Omega, \psi, \varphi\rangle$, a path through the model is: $\langle (n_1, n_2), (n_2, n_3), \ldots, (n_{j-1}, n_j)\rangle$ where elements of the path are control flow links and each n_x in the path is distinct. We shall say $n_i \prec n_j$ iff there exists a path $\langle (n_i, n_{i+1}), \ldots, (n_{j-1}, n_j)\rangle$ where n_i precedes n_j.

Definition 2: Neighbor Function
Let $\text{Neighbor}_p(n_i) : N \rightarrow 2^N$ *be a function that returns the neighbor of any node* n_i *in a process* p.

$$\text{Neighbor}_p(n_i) = \{n_j | n_j \neq n_i \wedge \\ (((n_i, n_j) \in F) \vee (n_i \not\prec n_j \wedge n_j \not\prec n_i))\}$$

□

Definition 3: Trace
Given a process model $p = \langle N, F, l, \Omega, \psi, \varphi\rangle$*, a trace is a sequence* $\sigma = \langle n_1, n_2, ., n_m\rangle$ *where:*

- $|N| = |\sigma|$.
- *For each pair* $\langle n_i, n_{i+1}\rangle$ *in* σ, $n_{i+1} \in \text{Neighbor}_p(n_i)$ *for* $1 \leq i \leq m$.
- *For any two nodes* $n_i, n_j \in \sigma$*, where* $i < j$, $n_j \not\prec n_i$.

The set of all traces of a process model is Σ_p. □

For any two nodes n_i and n_j in a trace σ, if $\sigma = \langle \ldots, n_i, \ldots, n_j, \ldots\rangle$ we say n_i comes before n_j. If for all traces $\sigma \in \Sigma_p$ of a given process model n_i comes before n_j then we say that n_i always comes before n_j and denote this $n_i \ll_\Sigma n_j$.

Cumulative effects and QoS values are computable over process traces. Accumulation of effects and QoS values have been discussed in previous work (Morrison et al., 2012; Hinge et al., 2009; Hoesch-Klohe et al., 2010; Koliadis & Ghose, 2006), the result of which is a consistent set of effects and QoS values that give a semantic meaning to a process model. An accumulation function *accumulate* takes as input a trace σ and returns a tuple $\langle \Upsilon, \mathcal{E}, \Gamma\rangle$ which are the set of cumulative QoS values, the set of cumulative effects and the cumulative customer state respectively. For details of accumulation the reader is referred to (Morrison et al., 2012; Hinge et al., 2009; Hoesch-Klohe et al., 2010; Koliadis & Ghose, 2006). The accumulation of the decision free process, from *[Receive complaint]* to *[end event]*

A process model with traces, accumulated effects and QoS values is represented with a case sequence.

Definition 4: Case Sequence
A case sequence C is a tuple:

$$\langle \mathcal{M}, \Pi, \mathcal{A}, N, F, l, a, \Omega, \psi, \varphi \rangle$$

Where:

1. *$\mathcal{M}$ is a set of processes that are part of the case sequence.*
2. *Π is a sequence of traces $\Sigma_p \in \mathcal{M}$.*
3. *$\mathcal{A}$ is a set of tuples $\langle \Upsilon, \mathcal{E}, \Gamma \rangle$ of QoS values, accumulated effects, and customer environment.*
4. *N is a finite non-empty set of nodes. ψ, φ are the start and end nodes respectively and $\psi, \varphi \in N$.*
5. *F a set of control flow links, $F \subseteq N \times N$.*
6. *Ω is a set of labels, each label ω is of the form $\langle$type, name, $\langle \upsilon, \epsilon, \gamma \rangle\rangle$. Where* type *is the type of element.* name *is the name of the element. $\langle \upsilon, \epsilon, \gamma \rangle$ is a tuple representing the QoS capabilities for the element, effect scenario of the element and customer state respectively.*
7. *$a : \Pi \rightarrow \mathcal{A}$ is function that maps trace sequences to accumulation tuples.*
8. *$l : N \rightarrow \Omega$ is a labelling function that assigns labels to nodes of the case sequence.*
9. *$\forall n \in N, (n, \psi) \notin F \wedge (\varphi, n) \notin F$ i.e., the start node has no incoming edges and the end node has no outgoing edges* □

For any case sequence C, if all $p \in \mathcal{M}$ are in a process portfolio $\mathcal{P}$ then we say that C belongs to $\mathcal{P}$,denoted $C \sqsubset \mathcal{P}$. To compute the accumulated effects and QoS capabilities for a case sequence, each sequential pair of traces $\langle \sigma_i, \sigma_j \rangle \in \Pi$, where $\langle \Upsilon_i, \mathcal{E}_i, \Gamma_i \rangle$ are the cumulative QoS, effect values and customer states for trace σ_i and $\langle \upsilon_{j_\psi}, \epsilon_{j_\psi}, \gamma_{j_\psi} \rangle$ are the QoS, effects, and customer states for the start node of σ_j first *pairwise-accumulate* $\langle \Upsilon_i, \mathcal{E}_i, \Gamma_i \rangle$ *and* $\langle \upsilon_{j_\psi}, \epsilon_{j_\psi}, \gamma_{j_\psi} \rangle$ then accumulate across the remains of the trace σ_j, the reader is referred to (Hinge et al., 2009) for methods of pairwise-accumulating cumulative effects with effects. The result of this process is a cumulative effect $\langle \Upsilon_P, \mathcal{E}_P, \Gamma_P \rangle$.

Returning to our example, if we wanted to determine the case wide effect for a customer who had phoned VirgaFone to make a complaint but due to staffing issues was unable to be verified. We would first create a decision free process of the path that the customer had taken through the process. Then compose the process with any other processes that the customer may have followed as part of their engagement with VirgaFone. A cumulative semantic effect of *{unhappy(customer), ¬ engaged(customer), ¬ resolution(customer)}* can be found. By computing QoS based artefacts such as engagement time we can compute the QoS effect, e.g. *{10min, $-10 staffing}*. Finally a case based effect could also be carried through *{80% churnRisk, -$50 goodwill}*

When describing an adaptive case management system an a difficulty exists of describing cases and caseflow across differing levels of abstraction (between perennial non-operational soft workflows and transient processes). In this framework, we propose that composing processes to correct the degree of abstraction is the best way to address this issue. Composition has been choosen here because the decomposition requires a pool of potential predefined subprocesses or atomic tasks (or requires extensive insightful thoughts by an analyst). Further to this, in the event of an organizational change, or in a change of understanding, transforming

the purpose of multiple layers of processes and their decomposition becomes cumbersome and expensive. An unrefined process is compound, and due to this that it is often necessary to consider the sequencing of processes or parallel execution of processes. When fused together, these composed processes may create a complex enough expression to consider the expression a case.

Given any number of process models with the purpose of investigating the consequences of running them in sequence or parallel, it becomes necessary to understand both the QoS value and semantic effect of the composed processes to gain insights. We compute accumulation over the composed process by manipulating processes and case sequences using a sequential combination operator and a parallel combination operator which both translate the composed processes into case sequences which we have already shown an accumulation procedure for. We will denote pairwise case sequence composition accumulation as $C_i \Cup C_j$. Sequential process composition, requires the selection of a trace from the set of possible traces for both of the processes. The end event of the first process is then converted to an intermediate event and joined to the start (converted to an intermediate event) of the selected trace from the next process in a series. Each new sequence of nodes shows an end-to-end arrangement of a composed process. Using pairwise accumulation along each sequence it is possible to compute the of effects and QoS values for the composed process. The joining of two processes in sequence is denoted by the operator $\otimes$.

To find parallel traces, we essentially join two processes into a parallel design. Converting the start and end events from each into intermediate events and placing the processes between parallel gateways. Using the methods in definition 3 to compute the set of traces for the composed process it is easy to find the end-to-end arrangement of the composed process. The joining of two processes in parallel is denoted by the operator $\oplus$.

Due to the nature of the sequential composition, in a case sequence composition description $\oplus$ has precedence over $\otimes$, i.e. $p_1 \otimes p_2 \oplus p_3 = p_1 \otimes (p_2 \oplus p_3) \neq (p_1 \otimes p_2) \oplus p_3$. The composed models generated through these procedures is a case sequence.

Given a process portfolio $\mathcal{P}$, we assume there exists a number of case sequences $C_1, C_2, \ldots$ that can be constructed by composing various processes and case sequences together. An accumulation function *accumulate* provides a method for semantic definition from each C_i to some tuple $a = \langle \Upsilon, \mathcal{E}, \Gamma \rangle$, i.e. *accumulate(C)* $= a$ where a is a closed wff, and due to the nature of effect accumulation $Th($*accumulate(C)*$) =$ *accumulate(C)*.

Let a *base case sequence* $B_{\mathcal{P}}$ for a particular process portfolio $\mathcal{P}$, be any case sequence $C \sqsubset \mathcal{P}$ with $U = \{p | p \in \mathcal{M}_C\}$ where there does not exist another case sequence $C' \sqsubset \mathcal{P}$ with $V = \{p | p \in \mathcal{M}_{C'}\}$ where $V \subset U$.

Given a set of base case sequences $\{B, B' \ldots \in \mathcal{B}_{\mathcal{P}}\}$, an *extension case sequence* of a base case sequence B with $W = \{p | p \in \mathcal{M}_B\}$, is any case sequence $C \sqsubset \mathcal{P}$ and $C \notin \mathcal{B}_{\mathcal{C}}$ with $U = \{p | p \in \mathcal{M}_C\}$ where $W \subset U$ and there does not exist another case sequence $C' \sqsubset \mathcal{P}$ and $C' \notin \mathcal{B}_{\mathcal{P}}$ with $V = \{p | p \in \mathcal{M}_{C'}\}$ where $V \subset U$ and $W \subset V$ and there exists an operator $\oplus$ or *otimes* where $B \oplus B' = C$ or $B \otimes B' = C$. A child extension is some case sequence C'^* that can be found by forming a chain of extension case sequences, i.e. if C is a base case sequence and C' is an extension case sequence for C, and C'' is an extension case sequence for C' etc., until C'^*

Definition 5: Case Sequence Accumulation
Let q $= \langle \Upsilon, \mathcal{E}, \Gamma \rangle$ *be a tuple describing some set of QoS and effect scenarios and* C *be a case sequence. Let some* base case sequence C_0 *have an associated* q' *, where* $q' \not\models q$ *and for each* $i \geq 0$ *where* C_{i+1} *is an* extension case sequence *of* C_i*:*

$accumulate(C_{i+1}) = Th(accumulate(C_i)) \Cup C_{i+1}$

Then q *is the semantic description and cumulative effect of* C *iff:* $q = \Cup_{i=0}^{\infty}(C_i)$ *and* C *is a child extension of* C_0*.* □

By finding case sequences, and then using an accumulation function, it is easy to describe the effect scenarios or QoS capabilities of a multiple processes instance, i.e., if multiple processes were used to handle a particular case, then a QoS capabilities like time taken can be computed. It is also possible to use semantic effect annotations to provide contextual information on the state of a case at any given point during the processing of the case across any number of processes.

5 Correlating Cases and Strategies

By having existing process models that describe the capabilities of an organization it is assumed that the organization intends to use the processes to achieve its strategies. Realization of functional goals in this context is done by defining functional goals in operational terms. The way that we define functional goals in operational terms is by correlating them to processes. If a correlation between functional goals and processes is made, then this is a demonstration of goal realization.

Goal realization can be undertaken in various ways as shown by Letier and Ponsard (Ponsard et al., 2007; Letier & van Lamsweerde, 2002). In this work, we use entailment to represent the realization of strategies.

Our framework will be presented using a finitely propositional language $\mathcal{LANG}$ over a set of propositional letters $\mathcal{A} = \{\alpha, \beta, \gamma\}$, truth-functional connectives $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$, and the truth-functional constants $\top, \bot$. An interpretation of $\mathcal{LANG}$ is a function from $\mathcal{A}$ to $\{T, F\}$; Ω is the set of interpreations of $\mathcal{LANG}$. A model of a well formed formula X is an interpretation that makes X true.

In classic propositional logic entailment is a relation between two well formed formulas X and Y. Each formula we will use to demonstrate realization is a prime implicate. Let Γ be a theory, X and Y be wffs, then the antecedent formula X entails the consequent formula Y iff $X \neq \emptyset$ and every model of X under Γ is a model of Y, denoted $\Gamma, X \models Y$. A functional goal is realizable if a set of effect scenario entails it and if the set of effect scenarios is consistent.

Definition 6: Goal Realization (Letier & van Lamsweerde, 2002; Ponsard et al., 2007)
Given background knowledgebase Γ*, a set* $\{\epsilon_1, \ldots, \epsilon_n\}$ *of effects scenarios, and a goal* G*, then the goal is definable in terms of the effect scenarios and realizable in terms of the case sequences where the effect scenarios come from if:*

1. $\Gamma, \epsilon_1 \wedge \cdots \wedge \epsilon_n \models G$ *(completeness)*
2. $\Gamma, \epsilon_1 \wedge \cdots \wedge \epsilon_n \not\models \bot$ *(consistency)*

□

By showing that each functional goal in a strategy model has a process or a case sequence that realizes it, we are able to demonstrate that there is synergy between the operational and strategic sides of the organization.

5.1 Satisfaction of Optimization Objectives

An optimization objective in a strategy model is a optimization function that provides an ordering over a set of case sequences given a QoS preference. A c-semiring framework (Bistarelli, Montanari, & Rossi, 1997) can be used to adequately model such preference orderings. In (Hoesch-Klohe et al., 2010), a number of instances for green QoS were described using this framework.

Given a set of case sequences $\{C_1, C_2, \ldots, C_n\}$ where each case sequence has a pair $\langle \Upsilon_{P_i}, \mathcal{E}_{C_i} \rangle$ of QoS capabilities and semantic effect scenarios. The set of possible QoS capabilities for a process portfolio $\Upsilon_{\mathcal{P}}$ is $\{\Upsilon_{Ci} | C_i \sqsubset \mathcal{P}\}$.

For example, consider two case sequences C_i and C_j each with one instance of QoS Capability described below.

Case Sequence	Cost	Time
C_1	\$1.00	10 *minutes*
C_2	\$1.50	5*minutes*

The capabilties for C_1 are $\{[\text{cost}_{C_1}]\$1.00, [\text{time}_{C_1}]10min\}$ and the capabilties for C_2 are $\{[\text{cost}_{C_2}]\$1.50, [\text{time}_{C_2}]5min\}$. The set of possible QoS capabilties for the process portfolio are then:

$$\Upsilon_{\mathcal{P}} = \{[\text{cost}_{C_1}]\$1.00, [\text{time}_{C_1}]10min, [\text{cost}_{C_2}]\$1.50, [\text{time}_{C_2}]5min\}$$

Definition 7: Optimial Case Sequence
Given a set of case sequences $\{C_i, \ldots, C_j\}$ with a set of capabilities Υ made of all of the case sequence QoS values where $\forall \upsilon_{C_l} \in \Upsilon_{C_l}, \upsilon_{C_l} \in \Upsilon$ and a c-semiring $O_{type} = \langle A, \oplus, \otimes, \bot, \top \rangle$ describing a particular QoS preference type and preference ordering $\leq_{O_{type}}$, and a set of cumulative QoS capabilities $\Upsilon \subseteq A$, we denote more preferred QoS capabilities υ_k^{*type} (with resp. to the preference ordering $\leq_{O_{type}}$). An optimal case sequences C_k^{*type} is a case sequences with the best QoS value. □

Continuing our example, to find the optimal case sequence with respect to *time*, we would compute the *minimal time* capabiltiy $\upsilon_{C_i}^{*\text{time}} = [\text{time}_{C_2}]5min$. An aggregation of objectives may be substituted into the optimization `type`.

6 Alignment Between Cases and Strategies

Alignment between cases and strategies follows from the previous section discussing the realisation between cases and strategies. Alignment is the correlation of cases sequences to strategies and provides a system level understanding of how effective the case management systems choreographing the business operations is. An alignment framework previously described in detail in (Morrison et al., 2012) can be used to establish correlations between the case sequences of our system and the strategic landscape of the business as a whole.

Alignment between a process and a goal

Definition 8: Alignment of Case Sequence with Goals
A case sequence C with a set of effect scenarios $\mathcal{E}_C$ realizes a goal G, iff $\exists \epsilon_i \in \mathcal{E}_C$ where $\epsilon_i \models G$ and $\forall \epsilon_j \in \mathcal{E}_C$, $\epsilon_j \wedge G \not\models \bot$. We will write: $\mathcal{ALN}_C^G$, if this is the case.

□

Consider a set of case sequences $\{C_1, \ldots, C_n\}$ where $C_i \sqsubset \mathcal{P}$. Given a goal G, we want to determine if the process portfolio $\mathcal{P}$ is aligned to the goal G. We can use the following basic test: if $\exists C \sqsubset \mathcal{P}$ s.t. $\mathcal{ALN}_C^G$ then, $\mathcal{P}$ is aligned to G.

Alignment between a process and a plan

Definition 9: Alignment with Plans

Let a plan L be a sequence of goals $\langle G_1, \ldots, G_n \rangle$. For the plan to be completely realized by a process model (or process models) each pair of consecutive goals $\langle G_i, G_j \rangle$ in the plan must be realized. A plan is realized and aligned to a set of processes if all consecutive goal pairs in the plan are realized. Pairs of goals are realizable in the following ways:

1. *Let C_k be a case sequence with process $p_k \in \mathcal{M}_k$, and C_l be a case sequence with a process $p_l \in \mathcal{M}_l$, s.t. $\mathcal{M}_k/\{p_k\} = \{\emptyset\}$ and $\mathcal{M}_l/\{p_l\} = \{\emptyset\}$. Given a case sequence C_m with $p_k, p_l \in \mathcal{M}_m$ where for all $n_k \in N_k$ and $n_l \in N_l$, $n_k \ll_{\Sigma_C} n_l$, if C_k realizes G_i (but not G_j) and C_m realizes $G_i \wedge G_j$ then the case sequence C_m realizes the goal pair.*
2. *Given a base case sequence C_n, where there is an activity a with effect scenario ϵ_a, an activity b with effect scenario ϵ_b and $a \ll_{\Sigma_n} b$, if $\epsilon_a \models G_i$, $\epsilon_a \not\models G_i \wedge G_j$, $\epsilon_b \models G_i \wedge G_j$ and there is an end effect scenario of case sequence C_n that entails $G_i \wedge G_j$ then the case sequence C_n realizes the goal pair.*

If a plan L is realizable by a case sequence C then the case sequence is aligned to the plan, denoted $\mathcal{ALN}_C^L$. Consider a set case sequences $\{C_1, \ldots, C_n\}$ constructed from process models in a process portfolio $\mathcal{P}$. Given a plan $L = \langle G_1, \ldots, G_n \rangle$, the process portfolio $\mathcal{P}$ is aligned to the plan L if $\exists C \sqsubset \mathcal{P}$ s.t. $\mathcal{ALN}_C^L$ then, $\mathcal{P}$ is aligned to L.

□

Given a description of goal and plan alignment to business processes, if a business wishes to find which process models are optimially aligned to the strategies then optimization objectives must be used. To compute optimal alignment scenarios, given a functional goal G, and two case sequences C and $C\prime$ with alignment relationships $\mathcal{ALN}_C^G$ and $\mathcal{ALN}_{C'}^G$.

For example, consider an organizational optimization objective $\mathcal{O}$: *'minimize cycle time'* applied to a functional goal encouraging the use of vacation time. Case C may be a manual case sequence that requires the employee to submit leave request forms and find their own replacements, and case sequence C' may be an automated process that automatically selects replacement employees and stream lines the approval process. A QoS execution description for case sequence C may be *Time < 2 days*, and the QoS execution description for case sequence C' may be *Time < 2 hours*. Provided that there are no alternative QoS objectives, then the selection function will select process C' as being the optimal process to satisfy the goal.

Definition 10: Alignment with Optimization Objectives
Given a functonal goal G, an optimization objective $\mathcal{O}$, and two alignment scenarios $\mathcal{ALN}_C^G$ and $\mathcal{ALN}_{C'}^G$ where $C, C' \sqsubset \mathcal{P}$, then the optimal case sequence is the case sequence that is

optimal with respect to the optimization objective $C \Updownarrow^G_{\mathcal{P}} \mathcal{O}$. *This is the process aligned to the goal that is more preferred based on the optimization objective.*

Similarly, given a plan L, *where* $\mathcal{ALN}^L_C$ *and* $\mathcal{ALN}^L_{C'}$ *the optimal case sequence is the case sequence aligned to the plan that optimizes the optimization objective, denoted,* $C \Updownarrow^L_{\mathcal{P}} \mathcal{O}$.
□

Using the above definitions we are able to define a notion of strategic alignment of business processes.

Definition 11: Strategic Alignment
Let $\mathcal{P}$ *be a process portfolio,* S *be a set of strategies, made up of a set of goals* $\mathcal{G}$, *a set of plans* $\mathcal{L}$, *and a set of optimization objectives* $\mathcal{O}$. *We say that* $\mathcal{ALN}^S_{\mathcal{P}}$ *iff:*

1. *For each* $G \in \mathcal{G}$, $\mathcal{ALN}^G_{\mathcal{P}}$ *and*
   $\forall C \sqsubset \mathcal{P} \cdot (\epsilon_C \wedge G \not\models \bot | \epsilon_C \in \mathcal{E}_C)$
2. *For each* $L \in \mathcal{L}$, $\mathcal{ALN}^L_P$ *and*
   $\forall C \sqsubset \mathcal{C} \cdot (\epsilon_C \wedge L \not\models \bot | (\epsilon_C \in \mathcal{E}_C))$

□

To then compute optimal case sequences that achieve strategic requirements we define optimal strategic alignment.

Definition 12: Optimal Strategic Alignment
Given a set of case sequences $\mathrm{OPT} = \{C | C \sqsubset \mathcal{P}\}$ *is aligned optimally to the set of strategies, denoted* $\mathcal{ALN}*^S_{\mathrm{OPT}}$ *iff:*

1. $\mathcal{ALN}^S_{\mathcal{P}}$
2. *For each* $G \in \mathcal{G}$, *there is a case sequence* $C \in \mathrm{OPT}$ *s.t.* $\mathcal{ALN}^G_C$ *and* $C \Updownarrow^G_{\mathcal{P}} \mathcal{O}$
3. *For each* $L \in \mathcal{L}$, *there is a case sequence* $C \in \mathrm{OPT}$ *s.t.* $\mathcal{ALN}^L_C$ *and* $C \Updownarrow^L_{\mathcal{P}} \mathcal{O}$
4. *There is no smaller set of case sequences that are optimal with respect to* S.

□

Diagnosis is often described as an abduction problem in AI. It's idea is to produce an explanation that best accounts for a set of observable symptoms. More precisely, a diagnostic conclusion should plausible enough to explain the symptoms and it should be significantly better than any other explanations. Abductive reasoning is used in many AI problems as a reasoning paradigm. Abduction, or Abductive inference is a form of inference that takes a set of observations, a set of possible events that could have occurred and then returns the most likely explanation for the observation(Josephson, 1994). It can be best described as a kind of interpreting inference. The role of abduction has been demonstrated in various applications, most commonly in problems that require diagnosis. It has been proposed as a reasoning paradigm in AI for planning, default reasoning and diagnosis (Kakas & Sadri, 2002). Abduction within the realm of first order logic can be defined with the following schema: For any given observation O, given T, a collection of facts, E is an explanation for T such that no other E' can explains T better than E does. Therefore E is the most likely explanation of events that led to O (Josephson, 1994).

Using an alignment framework as shown above, along with an abductive reasoner, it is possible to find potential best set of processes to run together dynamically, using the

organisational strategies, existing semantically annotated processes and execution level case state information.

7 Related Work

In (Khomyakov & Bider, 2000) Khomyakov and Bider propose a reverse approach to achieving case flexibility by first describing a set of workflow states and then using restrictions of obligations, prohibitions and recommendations a theoretical hybrid automata machinery is able to constructively create all combinations of states and hence describing all that is possible given a set of process and a set of constraints. In (Mundbrod et al., 2012) Mundbrod et. al. has presented a lifecycle methodology and framework for supporting collaborative knowledge work. Mundbrod has identified a large number of elements that can be used to describe large scale and complex systems including complex financial services and criminal investigation scenarios, which involve highly trained knowledge workrs.

Hildebrandt et. al. (Hildebrandt et al., 2011b, 2011a) have approached the creation of a dynamic and declarative case management system with a system of Dynamic Condition Response Graphs that provide state transition modeling for case systems. In particular their work focuses on the execution level store of case models and have provided a rigorous and formal model of case management systems. We believe that their condition response graphs are complementary to our Case Sequence models. Using our framework a case management system can be designed at a broad overviewing level; and then the Dynamic Condition Response Graph can be used to model and assess case behaviour at execution time. In future work we would like to provide further evaluation of our framework under design usage to compare with the execution support of the Hildebrandt model.

In previous work (Van Der Aalst, 1998; Dijkman & Van Gorp, 2011; Bose & van der Aalst, 2010; Rinderle-Ma, Reichert, & Weber, 2008; Ly et al., 2012), many researchers have described formal models for graph encoding specific process model types. In our work we provide a general summary for process model formulation (based loosely on the work of our peers); however, we draw attention to the fact that all automation and computation in our framework is done at the design level rather than at the execution level and as such various elements of some graph encodings do not fit with the definitions we've provided. By maintaining a process and task definition at the design level, reconfiguration can be computed without the need for execution, so that new adaptive task sequences can be constructed without an example execution trace. For example, in (Bose & van der Aalst, 2010), Bose and van der Aalst et. al. have described an execution trace over the state space of unbounded logs for a workflow net, where their notion of an ordering relation is a sequence of possible state transitions. Our notion differs as it is done at a design level. Further to this, each trace dealt with in this paper through a semantically annotated process model is one of the many possible interleaving executional designs that exists in the process model. In (Rinderle-Ma et al., 2008; Ly et al., 2012) Rinderle-Ma et. al. have described a notion of executional event trace that is similar to the executional traces of van der Aalst, this notion of trace like those in (Bose & van der Aalst, 2010) requires dropping from design into the domain of execution artifacts.

8 Conclusion

In this chapter, we have provided a method to compute case sequences from a collection of process models. Elements on the system described have been developed into a prototype library[3]. The result and benefit of using a case management system formed from existing legacy process management systems is that transition and change costs will be dramatically reduced for the organisation. The results of moving towards adaptive case management using our framework will provide organizational case managers a apparatus to understand the current case state of affairs across the entire operational context. The framework that we have presented contributes to a better understanding of adaptive case management and further tool support will equipped decision makers with a device to understand sustainability of this technology in an operational context.

References

Bistarelli, S., Montanari, U., & Rossi, F. (1997). Semiring-based constraint satisfaction and optimization. *Journal of the ACM (JACM)*, *44*(2), 201–236.

Bleistein, S. J., Cox, K., & Verner, J. (2006). Validating strategic alignment of organizational it requirements using goal modeling and problem diagrams. *Journal of Systems and Software*, *79*(3), 362 – 378.

Bose, R. J. C., & van der Aalst, W. (2010). Trace alignment in process mining: opportunities for process diagnostics. In *Business process management* (pp. 227–242). Springer.

Dijkman, R., & Van Gorp, P. (2011). Bpmn 2.0 execution semantics formalized as graph rewrite rules. In *Business process modeling notation* (pp. 16–30). Springer.

Edirisuriya, A., & Johannesson, P. (2009). On the alignment of business models and process models. In *Business process management workshops* (Vol. 17, p. 68-79).

Ghose, A. K., Lê, L., K, H.-K., & Morrison, E. D. (2010). The business service representation language: A preliminary report. In *Service modelling and representation techniques.*

Gordijn, J., Soetendal, J., & Paalvast, E. (2005). V A^3: Governance selection in value webs. In *Challenges of expanding internet: E-commerce, e-business, and e-government* (Vol. 189, p. 17-31).

Hildebrandt, T., Mukkamala, R. R., & Slaats, T. (2011a). Declarative modelling and safe distribution of healthcare workflows. In *First international symposium, fhies.*

Hildebrandt, T., Mukkamala, R. R., & Slaats, T. (2011b). Designing a cross-organizational case management system using dynamic condition response graphs. In *15th international enterprise distributed object computing conference.*

Hinge, K., Ghose, A. K., & Koliadis, G. (2009). Process seer: A tool for semantic effect annotation of business process models. In *13th ieee international edoc conference.*

Hoesch-Klohe, K., Ghose, A. K., & Le., L.-S. (2010). Towards green business process management. In *Proc. of the ieee international services computing conference.*

Josephson, S. G. (1994). *Abductive inference: computation, philosophy, technology* (J. R. Josephson & S. G. Josephson, Eds.). Cambridge University Press.

Kakas, A. C., & Sadri, F. (Eds.). (2002). *Abduction in logic programming, computational logic: Logic programming and beyond.* Springer Berlin Heidelberg.

[3] The source code for the framework can be found online at `http://www.dsl.uow.edu.au/~edm92/textseer`

Khomyakov, M., & Bider, I. (2000). Achieving workflow flexibility through taming the chaos. In *Oois 2000 - 6th international conference on object oriented information systems.*

Koliadis, G., & Ghose, A. (2006). Relating business process models to goal-oriented requirements models in kaos. In *Advances in knowledge acquisition and management* (Vol. 4303, p. 25-39).

Koliadis, G., Ghose, A., & Padmanabhuni, S. (2008). Towards an enterprise business process architecture standard. In *Ieee congress on services* (pp. 239–246).

Letier, E., & van Lamsweerde, A. (2002). Deriving operational software specifications from system goals. *SIGSOFT Softw. Eng. Notes*, *27*(6), 119–128.

Ly, L., Rinderle-Ma, S., Göser, K., & Dadam, P. (2012). On enabling integrated process compliance with semantic constraints in process management systems. *Information Systems Frontiers*, *14*(2), 195-219. Retrieved from `http://dx.doi.org/10.1007/s10796-009-9185-9` doi: 10.1007/s10796-009-9185-9

Morrison, E. D., Ghose, A. K., Dam, H. K., Hinge, K. G., & Hoesch-Klohe, K. (2012). Strategic alignment of business processes. In *Service-oriented computing-icsoc 2011 workshops* (pp. 9–21).

Mundbrod, N., Kolb, J., & Reichert, M. (2012). Towards a system support of collaborative knowledge work. In *1st international workshop on adaptive case management.*

Ponsard, C., Massonet, P., Molderez, J., Rifaut, A., Lamsweerde, A., & Van, H. (2007). Early verification and validation of mission critical systems. *Formal Methods in System Design*, *30*, 233-247.

Rinderle-Ma, S., Reichert, M., & Weber, B. (2008). Relaxed compliance notions in adaptive process management systems. In Q. Li, S. Spaccapietra, E. Yu, & A. Olivé (Eds.), *Conceptual modeling - er 2008* (Vol. 5231, p. 232-247). Springer Berlin Heidelberg. Retrieved from `http://dx.doi.org/10.1007/978-3-540-87877-3_18` doi: 10.1007/978-3-540-87877-3_18

Strijbosch, K. (2011). *Adaptive case management: A new way of supporting knowledge work* (Unpublished doctoral dissertation). Radboud Universiteit Nijmegen.

Van Der Aalst, W. M. P. (1998). The application of petrinets to workflow management. *Journal of Circuits, Systems and Computers*, *08*(01), 21-66.

van Lamsweerde, A. (2001). Goal-oriented requirements engineering: a guided tour. In *Requirements engineering, 2001. proceedings. fifth ieee international symposium on* (p. 249 -262).

Wang, H., & Ghose, A. K. (2010). Green strategic alignment: Aligning business strategies with sustainability objectives. In B. Unhelkar (Ed.), *Handbook of research in green ict* (pp. 29–41).

Zhu, W.-D., Kirchner, M., Ko, T., Oland, M., Prasad, B., Prentice, M., & Ruggiero, M. a. (2013). *Advanced case management with ibm case manager.* IBM Redbooks.