

Combining i^* and BPMN for Business Process Model Lifecycle Management

George Koliadis¹, Aleksandar Vranesevic¹, Moshiur Bhuiyan¹, Aneesh Krishna¹,
and Aditya Ghose¹

¹School of Information Technology and Computer Science (SITACS),
University of Wollongong (UOW), Gwynneville NSW 2522, Australia
{gk56, av85, mmrb95, aneesh, aditya}@uow.edu.au

Abstract. The premise behind ‘third wave’ Business Process Management (BPM¹) is effective support for change at levels. Business Process Modeling (BPM²) notations such as BPMN are used to effectively *conceptualize* and *communicate* process configurations to relevant stakeholders. In this paper we argue that the management of change throughout the business process model lifecycle requires greater conceptual support achieved via a combination of complementary notations. As such the focus in this paper is on the co-evolution of operational (BPMN) and organizational (i^*) models. Our intent is to provide a way of expressing changes, which arise in one model, effectively in the other model. We present constrained development methodologies capable of guiding an analyst when reflecting changes from an i^* model to a BPMN model and vice-versa.

1 Introduction

Business process models play a key role in both organizational management [1] [2] and enterprise information systems development [3]. Many notations developed for the task of modeling business processes, have their own focus of *application* and appropriate *audience* [4] [5] [6] [7] [8]. High-level conceptual models provide an understanding of an organization from an intentional and social perspective [9] for reasoning support during redesign [9]. In comparison, lower-level technical models are especially suited for applications in the description, execution and simulation of business processes [8].

Business process development should be based on principled high-level models of the enterprise and the business context. Commonly, processes are formulated in an ad-hoc fashion without reference to these high-level models. Some of the most prominent modeling notations enlisted are focused towards technically-oriented data, and process modeling notations such as ER, Data-Flow, Systems Flowcharting and UML and workflow modeling [10]. In this work, we offer constrained development methodologies to guide development of process models from higher-level conceptual models. This supports life-cycle management in the following sense: when changes occur to the high-level model, these can be reflected in the process model, and vice-

versa. In this paper, Section 2 provides a background to business process modeling with an overview of our chosen notations. Section 3 illustrates concepts/methods provided in our methodologies (with examples). The paper is concluded in Section 4.

2 Background

The notations used for modeling business processes have been categorized in many works, based on their conceptual features [4] [5] [6] [7] [8]. The common principle recognized in all analyses is that some notations are more suited towards specific *audiences* (i.e. with either technical/non-technical backgrounds) or *applications* (i.e. possibly for description, re-design or execution) throughout the business process lifecycle. Many notations focus on specific aspects, with limited relation/traceability to other important business process aspects. This has brought about the need for an enterprise view [6] to support the development and maintenance of rich models that provide an enhanced ability to *conceptualize*, *communicate* and *understand* business processes, and their context of operation.

In related work, some preliminary ideas in [11] have been proposed for developing a BPMN model given the existence, and agreement to, an i^* model of the process. Six steps are provided for mapping between constructs, with no consideration for reflecting change and consistency made. Also, an approach for deriving a BPMN model from a business model is proposed in [12], achieved through the intermediate translation of the business model into an activity dependency model that can then be translated into a business process model. In this work, we provide a simpler approach aimed at reducing added complexity and/or misinterpretations during modeling. Furthermore, much work has been completed on supporting guided translation and co-evolution of i^* into various other behavioral modeling notations and languages [13] [14] [15]. The primary aim in these approaches is to further develop detailed design artifacts that can lead onto implemented systems, or directly be used in the configuration of agent-based systems. However, our primary focus is on modeling lifecycle support during BPM¹ projects whereby the concern is for the development and/or assessment of detailed business process designs. The work in this paper extends previous work in [16]. In comparison to previous work, we take the following approach to lifecycle management: when changes to a business process model (i.e. BPMN – [17]) occur, these changes must ensure some notion of consistency with a higher-level enterprise model, and vice versa. In this instance, an i^* model [9].

2.1 Agent-Oriented Conceptual Modeling (AOCM) with i^*

i^* supports modeling rich organizational contexts by offering high-level social/anthropomorphic abstractions (such as goals, tasks, soft goals and dependencies) as modeling constructs for reasoning support during business process redesign [9] [7]. Figure 1 represents a simple i^* *Meeting Scheduling* model. The central concept in i^* is that of intentional actor. These can be seen in the Meeting Scheduling model as nodes representing the intentional/social relationships between three (3) actors

required to schedule a meeting: a *Meeting Initiator* (MI); *Meeting Scheduler* (MS); and, *Meeting Participant* (MP).

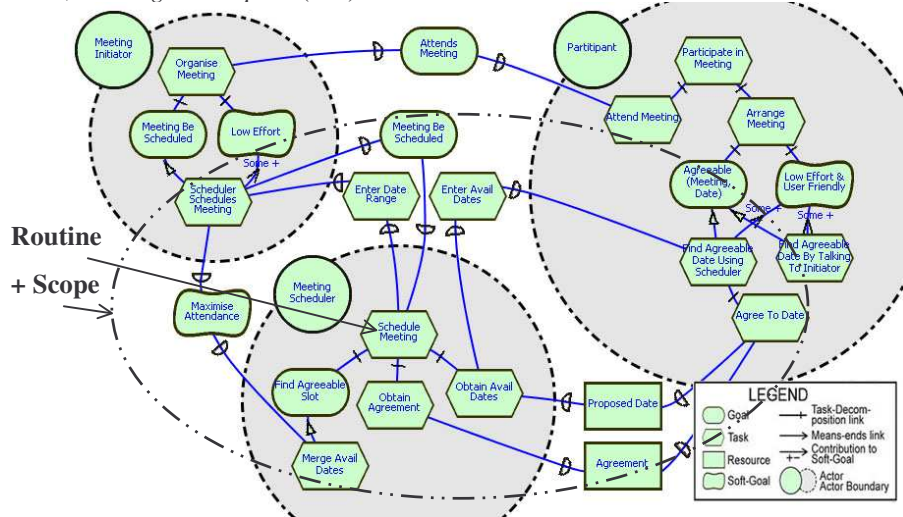


Fig. 1. An *i** Strategic Rationale (SR) Meeting Scheduling Model with a Routine Illustrated

The *i** framework consists of two modeling components [9] Strategic Dependency (SD), and Strategic Rationale (SR) models. The SD model consists of a set of nodes and links. Each node represents an actor, and each link between the two actors indicates that one actor depends on the other for something in order that the former may attain some goal. The depending actor is known as *dependor*, while the actor depended upon is known as the *dependee*. Dependencies may involve *goals* to be achieved (e.g. *MeetingBeScheduled*), *tasks* to be performed (e.g. *EnterAvailDates*), *resources* to be furnished (e.g. *Agreement*), or *soft-goals* (optimization objectives or preferences) to be satisfied (e.g. *MaximizeAttendance*).

The SR mode further represents internal motivations and capabilities (i.e. processes or routines) accessible to specific actors that provide illustration of how dependencies can be met. In *i**, a *routine* [9] specifies an intended course of action an actor may pursue given a set of alternatives. These elements and their relationships represent the strategic requirements of a business process when invoked in a specific context. For example, to *ScheduleMeeting* (illustrated in Figure 1 with its Scope) that includes three sub-tasks and six dependencies with two additional actors. Tasks in *i** may be primitively workable whereby the actor responsible for the element believes that it can achieve its requirements at execution time – i.e. it is sufficiently reduced during decomposition. In comparison to BPMN however, a primitively workable element may still be represented as a sub-process as the term does not imply a ‘primitively executable action’ (i.e. application of analyst / designer discretion). Furthermore, for a routine to be workable, all involved actors must be committed to satisfying their dependencies [9].

The Tropos project [18] aims to provide methodological support for advancing the *i** framework further towards architectural and detailed design where dynamic / be-

havioral aspects are of importance. Specifically, Formal Tropos (FT) – see [19], is a part of the Tropos project that provides a specification language for modeling dynamic aspects of an i^* model via formal annotation of *Creation* and *Fulfillment* conditions. These conditions are specified using first-order typed linear temporal logic and prescribe the constraints on an elements lifecycle. In this work, we take the same approach to annotation (with the use of fulfillment conditions annotated to i^* models). In comparison, our work is illustrated via informal annotations.

2.2 Business Process Modeling with BPMN

The Business Process Modeling Notation (BPMN), developed by the Business Process Management Initiative (BPMI.org) [17] is primarily a technically-oriented business process modeling notation that supports the assignment of activity execution control to entities within an organization via ‘swim-lanes’. BPMN has the capability to map directly to executable process languages including XPDL [20] and BPEL [17] [21]. Furthermore, an analysis of BPMN [22] also stated its high maturity in representing concepts required for modeling business process, apart from some limitations in terms of representing state, and the possible ambiguity of the swim-lane concept.

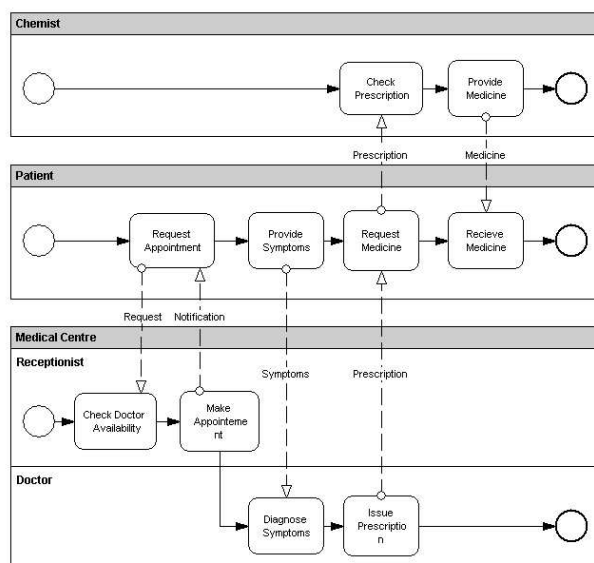


Fig. 2. A BPMN Patient Treatment Business Process Model

Figure 2 represents a simple BPMN Patient Treatment process. Processes are represented in BPMN using **flow nodes**: *events* (circles), *activities* (rounded boxes), and *decisions* (diamonds); **connecting objects**: *control flow links* (unbroken directed lines), and *message flow links* (broken directed lines); and **swim-lanes**: *pools* (high-level rectangular container), and *lanes* partitioning pools. These concepts are further discussed in [17].

3 Constrained Development Methodologies

We propose constrained development methodologies to guide the derivation or maintenance of one type of model given the availability of the other. The development is supported with the introduction of two concepts: *fulfillment conditions* (i.e. as in [19]) and *effect annotations*.

An *effect* is broadly defined as the result (i.e. product or outcome) of an activity being executed by some cause or agent. An *effect annotation* is a specific statement relating to the outcome of an activity, associated to a *state altering construct* in a given model. During BPM², effects are annotated to atomic tasks/activities or sub-processes within an actor's lane. The execution of a number of activities in succession results in a *cumulative effect* that includes the specific effects of each activity in the sequence. We also note the fact that certain effects can undo prior effects (i.e. in the case of compensatory activities). Effect annotations may possibly be formalized using the formal layers of some currently well-developed Goal-Oriented Requirements Engineering (GORE) methodologies [23] [19], however, we only state their applicability in this work, and aim towards possible integration in the future.

Fulfillment conditions are annotated to *tasks* and *goals* assigned to actors in an SR diagram, and *dependencies* (i.e. not including *soft-goals* as these are used during assessment of alternatives and describe non-functional properties to be addressed) in an *i** model. A *fulfillment condition* [19] is a statement specifying the required conditions realized upon completion of a given task, goal or dependency. Fulfillment conditions recognize the required effects on a business process model. For example, a fulfillment condition for a task dependency to *EnterADateRange*, may be the *DateRangeCommunicated* effect (subsequently required by the task assigned to a dependee actor).

3.1 Annotation and Propagation

Tasks, goals and dependencies are annotated with fulfillment conditions in an *i** model. Additionally, the tasks assigned to participants in a BPMN model are annotated with effects for assessment against fulfillment conditions.

Tasks associated to dependencies on the *dependee* side may require additional effects when related to a BPMN model. That is, the fulfillment conditions for a dependency may not be explicitly stated against the tasks. For example, the fulfillment condition for *ProposedDateProvided* (i.e. annotated to the *ProposedDate* resource dependency in Figure 1) will be propagated to the *ObtainAvailDate* task. This should occur during annotation, whenever a fulfillment condition is annotated to a *resource*, *goal* or *task* dependency.

Effect annotations in BPMN models are propagated via *trajectories*. A trajectory is a sequential execution of activities terminating at an end state that represents the operational goal of the process. Control flow links between events, activities, and gateways within a BPMN model indicate the flow of trajectories. Effects within a process are accumulated during forward traversal through a trajectory. This accumu-

lation ensures that any compensatory activities, that may undo effects, are also taken into account during traversal.

3.1.1 Annotating the Meeting Scheduling Model (Figures 1 and 4).

Table 1. Annotation of Fullfillmment Conditions to Respective Tasks/Dependencies

Task/Dependency (Figure 1)	Fulfillment Conditions	Task Annotation (Post Development – Figure 4)
MI: SchedulerSchedules Meeting	DateRangeEnteredIntoScheduler; DateRangeCommunicatedToScheduler	1; 1;
MS: ScheduleMeeting	AgreedDateKnownToInitiator	4
MS: ObtainAvailableDates	ProposedDateProvided; AvailableDatesObtained; AvailableDatesStored; AvailableDatesValidated	2 (message); 2; 2; 2
MS: ObtainAgreement	AgreementObtained; AgreementRecorded	4; 4
MS: MergeAvailableDates	AvailableDatesMerged	3
P: AgreeToDate	DateAgreedTo; AgreementProvided;	6; 6 (message)
P: FindAgreeableDateUsing Scheduler	AvalDatesEnteredIntoScheduler; AgreeableDateFoundUsingScheduler	5; 6
MS-Dep->MI: EnterDateRange	DateRangeCommunicatedToScheduler	1
MI-Dep->MS: MeetingBeScheduled	AgreedDateKnownToInitiator	4
MS-Dep->P: EnterAvailDates	AvailDatesEnteredIntoScheduler	5
P-Dep->MS: ProposedDate	ProposedDateProvided	2
MS-Dep->P: Agreement	AgreementProvided	6 (message)

3.2 Scope Projection

In order to evaluate consistency between the two notations, we provide some rules for projecting the scope of the i^* model. In the current case, i^* models are likely to represent a broader scope in comparison to a specific BPMN model as they are applied to capture the greater organizational context. Scope projection is based on an identification of the business process (represented in BPMN) as a *routine* assigned to an actor in an i^* model.

- *Rule 1:* The root node of the routine traceable to the process in consideration and all tasks in its first level of decomposition from are to be within scope.
- *Rule 2:* All dependencies that are associated to a task within the scope of the routine, where the actor in control of the routine (initiator) is the depender are within the scope of the process; as well as the tasks assigned to dependee actors.
- *Rule 3:* All dependencies that are associated to a task within the scope of the routine, where the initiator is the dependee are within the scope of the process iff the task assigned to the depender is part of some decomposition of a task in the scope of the process as per Rule 2; as well as the tasks assigned to the depender actors.

3.3 Consistency Evaluation

We introduce consistency rules to provide a mechanism for ensuring consistency between i^* and BPMN models (developed with consideration to [19]).

- *Rule 1:* Every actor in an i^* model required as a participant in the routine (traceable to the business process) and any of their tasks must be represented in the BPMN model (and vice versa), assessed via application of scope projection rules.
- *Rule 2:* There must exist a trajectory in the process model, whereby the operational objective (as encoded in the accumulated fulfillment conditions of traceable tasks) of the routine is achieved, and the sequence of activities is consistent with the requirements specified in the routine as further outlined below:
 - *Rule 2.1:* The accumulated effect of all tasks and goals traceable to the routine must achieve accumulated routine fulfillment conditions during forward traversal of at least one trajectory in the process model; AND,
 - *Rule 2.2:* The fulfillment of a task on the depender side of a dependency must not be realized before the fulfillment of the dependency upon accumulation of effects during forward traversal of the same trajectory.

3.4 Constrained Development of a Business Process Model given a High-Level Conceptual Model

These steps are based on the aforementioned consistency rules aimed towards providing analyst guidance during initial model development.

- *Step 1: Identify internal and external actors in i^* diagram.*
- *Step 2: Map elements to equivalent constructs within the BPMN model.* See sub-steps below.
 - *Step 2.1: Map Participants.* The greater organization for which the i^* model is represented is signified as a pool in BPMN. Any external participants are also represented as pools. Internal organizational actors are represented as lanes within the organizational pool.
 - *Step 2.2: Map Activities.* Tasks within i^* are represented as either sub-processes or atomic activities within BPMN assigned to actors within pools and lanes.
- *Step 3: Sequence required tasks/sub-processes and introduce control and sequence flow links by analyzing fulfillment conditions.* Tasks placed within each pool or lane are now sequenced to conform to routine requirements by taking Consistency Rule 2 (see: Section 3.3) into consideration. This requires that tasks be sequenced using control flow links in a manner that results in a trajectory satisfying fulfillment conditions on an i^* model. Control flow links are used to indicate realization of dependencies between actors within the same organization. In order to realize dependencies between organizational boundaries, a message flow link is used to represent the dependency going from the depender lane to the dependee lane. This may require single/multiple messages between tasks derived via analysis of fulfillment conditions.
- *Step 4: Elaborate on sub-processes.* The choice to introduce tasks or sub-processes into the BPMN diagram for specific tasks in the i^* model is made in *Step 2.2*. The analyst can develop each sub-process guided by the list of required fulfillment conditions annotated to the i^* task that the sub-process realizes.

Figure 3 illustrates the application of the constrained development methodology in the context of the Meeting Scheduling model represented in Figure 1, with annotations applied in Table 1. Much of the detail has been omitted for brevity. The following section describes a possible change requirement and its reflection within an i^* model for further analysis.

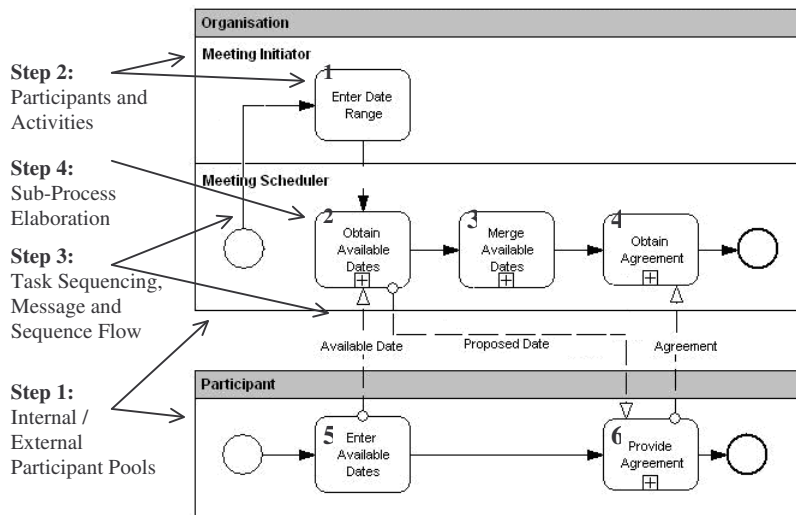


Fig. 3. BPMN Process Model derived using the Constrained Development Methodology

3.4.1 Reflecting Changes in an i^* Model to an associated BPMN Model.

The scope projection techniques are used to assess whether a change in an i^* model will impact a BPMN model. These guidelines aim to support the reflection of change between i^* and BPMN models for the specific instances of impacting change outlined below.

- *Step 1:* For each classification outlined below apply associated changes.
 - *Addition of an actor.* If a new actor has been added to the i^* model, a swimlane (i.e. for an internal actor) or pool (i.e. for an external actor) will need to be placed on the process model. Additionally, new dependencies must exist between the actor and existing actors (described below). These dependencies will be included for all new actors where the dependency is related to the routine and actor is the dependee. However, where the actor is the depender they will only be included if linked to a task in an existing dependency graph (see Scope Projection rules).
 - *Addition of a goal/task/resource dependency.* If a new dependency has been added to the i^* model, then this may require the addition of new activities/sub-processes and message flow links within the BPMN model (as described below).
 - *Addition of a goal or task.* The addition of a goal or task will require the addition of a task within the BPMN model. The addition of these tasks must be

scoped to their respective actors, and any dependencies must be realized via message-flow links where one of the actors is external to the organization.

- *Step 2:* Re-apply consistency rules to both models to assess whether consistency has been maintained.

Consider the following example applied to the *Meeting Scheduling* example in Figure 1 (*i**) and Figure 3 (BPMN). A new requirement within in the form of a task dependency between the *Meeting Initiator* (i.e. the dependee) and the *Meeting Scheduler* (i.e. the depender) to *ProvideParticipantPrioritization*. Participant prioritization means that the *Meeting Initiator* must now prioritize the current list of participants in order for the *Meeting Scheduler* to *MergeAvailableDates* and *FindAnAgreeableSlot* effectively.

Given the application of our approach for guiding an analysts decision, it can be inferred that the effect for *ParticipantPrioritizationProvided* will propagate within the *i** model as a fulfillment condition on the *SchedulerSchedulesMeetingTask*. Furthermore, given *Consistency Rule 3*, requires that *ParticipantPrioritizationProvided* occurs prior to the fulfillment of the *MergeAvailableDates* fulfillment conditions. This information can then be used to highlight the scope of change within the BPMN model to a point within a trajectory prior to the required effects of *MergeAvailableDates*, where an activity controlled by the initiator is able to realize the required effect.

3.5 Constrained Development of a High-level Conceptual Model given a Business Process Model

The following steps provide systematic guidance for developing an *i** model given an already existing process model. Figure 5, illustrates the constrained development of the *Patient Treatment* BPMN model in Figure 2.

- *Step 1: Map elements to equivalent constructs within the i* model.*
 - *Step 1.1: Map Participants.* Both pools and lanes in a BPMN model represent actors in an *i** model. These can be directly translated into the model.
 - *Step 1.2: Map Activities.* Represent activities and sub-processes as ‘primitively workable’ tasks assigned to actors in *i**.
- *Step 2. Apply intentional reasoning.*
 - *Step 2.1: Query the Intention of Tasks.* Intentional reasoning is applied to identify higher-level intentional elements and dependencies by querying the intention of tasks. This step aims to guide the further understanding and representation of an actors motivations.
 - *Step 2.2: Query the Intention of Flow-Links.* Analyze control and message flow between actor boundaries to identify goal, task and resource dependencies. These types of links can be used as a primary heuristic for identifying possible dependencies between actors.
- *Step 3: Identify soft-goal dependencies in the i* model.* The representation of soft-goals (including dependencies) are not in the scope of the BPMN notation.

3.5.1 Reflecting Changes in a BPMN Model to an associated i^* Model

These steps indicate how BPMN model change may be reflected in the i^* model:

- *Step 1:* For each classification of change, apply the following changes.
 - *Addition of a swimlane or pool.* If a swimlane or pool is added, then a new actor will be required within the i^* model. This will include the addition of new dependencies and tasks within the i^* model. A primary heuristic for identifying dependencies includes message flow links and control flow links between pools and lanes (message flow indicates a resource dependency for some information).
 - *Addition a task to an existing swimlane or pool.* If a new task is added to a swimlane or pool, this will require a task to be decomposed from the root node of the routine traceable to the current process.
- *Step 2:* Re-apply consistency rules assess whether maintenance.

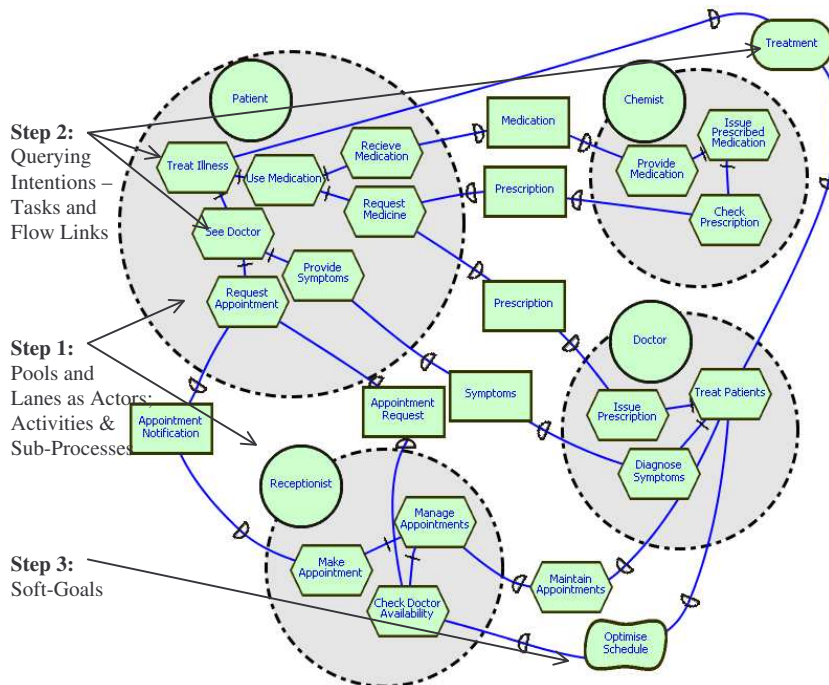


Fig. 4. An i^* 'Patient Treatment' Process

Consider now a scenario where the business process model is modified to improve the performance of the *IssuePrescription* task which has been identified to be a major operational bottleneck. The task is improved by including a task before hand which checks the patient's previous medical history to identify previous prescriptions for the patient for similar illnesses (e.g. common flu). We name the task *CheckPatientMedicalHistory*. Furthermore, the client is now encouraged to provide information on his medical background, which we represent as a task named *ProvideMedicalHistoryInformation*. We now proceed to add an additional task within the bounds of the Doctor agent and an additional task within the bounds of the Patient agent.

As in the previous case we use intentional reasoning to identify that the added task, within the Doctor agent, contributes to the higher level task of *TreatingPatients*. We apply the same technique to justify the placement of the *ProvideMedicalHistoryInformation* task as a decomposition task under the *RequestMedicine* task.

The added message flow in the BPMN diagram is represented as a resource dependency between the Patient and the Doctor, where the Doctor requires the Patient to provide his previous medical history. We also introduce the soft-goal between the Patient and the Doctor, titled *TimelyDrugPrescription*, indicating the fact that the Doctor will try to improve the time required to prescribe medication to the Patient.

4 Conclusion

In this work, we have illustrated an initial approach for supporting the lifecycle of business process models with the complementary use of *i** - a well developed notation for modeling organizational contexts, and BPMN – a newly developed notation for modeling business processes. The approach for reflecting changes in organizational context to changes in the design of business processes provides an effective mechanism for aligning business processes with organizational objectives. Similarly, operational improvements can be mapped back to organizational objectives to facilitate analysis and ensure no conflicts exist with existing objectives. Although these steps are preliminary we believe their systematic nature makes them available for automation in all phases, and are pursuing this task, through the development of a software tool, along with further refinement of the approach.

References

1. Smith, H. Fingar, P.: Business Process Management – The Third Wave. Meghan-Kiffer Press, Tampa Florida (2003)
2. Hammer, M. Champy, J.: Reengineering the Corporation: A Manifesto for Business Revolution. HarperBusiness, (1993)
3. Dumas, M. Aalst, W. M. P. and Hofstede, A. H.: Process-Aware Information Systems: Bridging People and Software Through Process Technology. Wiley-Interscience (2005)
4. Bider, I. Johannesson, P.: Tutorial on: Modeling Dynamics of Business Processes – Key for Building Next Generation of Business Information Systems. In: The 21st International Conference on Conceptual Modeling (ER2002), Tampere, FL, October 7-11 (2002)
5. Kavakli, V. and Loucopoulos, P.: Goal-Driven Business Process Analysis - Application in Electricity Deregulation. In: Information Systems, 24(3) (1999) 187-207
6. Loucopoulos, P. and Kavakli, E.: Enterprise Modeling and the Teleological Approach to Requirements Engineering. In: International Journal of Intelligent and Cooperative Information Systems 4(1) (1995) 45-79
7. Katzenstein G. Lerch F. J.: Beneath the surface of organizational processes: a social representation framework for business process redesign. In: ACM Transactions on Information Systems (TOIS), 18(4) (2000) 383-422

8. Yu, E.: Models for Supporting the Redesign of Organizational Work. In: Proceedings, Conf. on Organizational Computing Systems (COOCS'95) Milpitas, California, USA, August 13-16 (1995) 225-236
9. Yu, E.: Modelling Strategic Relationships for Process Reengineering. PhD Thesis, Graduate Department of Computer Science, University of Toronto, Toronto, Canada (1995) ~124
10. Davies, I, Green, P. Rosemann, M. Gallo, S.: Conceptual Modelling – What and Why in Current Practice. In: Lecture Notes in Computer Science, Volume 3288 (2004) 30-42
11. Cysneiros, L.M. Yu, E. Addressing Agent Autonomy in Business Process Management - with Case Studies on the Patient Discharge Process. In: Proc. of the 2004 Information Resources Management Association Conference, New Orleans, May, (2004)
12. Andersson, B. Bergholtz, M. Edirisuriya, A. Ilayperuma, T. Johannesson, P.: A Declarative Foundation of Process Models. In: Lecture Notes in Computer Science, Volume 3520 (2005) 233–247
13. Krishna, A., Ghose, A. K., Vranesevic, A.: Loosely-coupled Consistency between Agent-Oriented Conceptual Models and UML Sequence Diagrams. In: Special issue on Agent-Oriented Software Development Methodologies International Journal of Multi-Agent and Grid Systems. Accepted. (2006)
14. Dasgupta, A., Salim, F., Krishna, A., Ghose, A. K.: Hybrid Modelling using *i** and AgentSpeak (L) Agents in Agent-Oriented Software Engineering. To appear in: The Proceedings of 8th International Conference on Enterprise Information System (ICEIS-2006), Paphos, Cyprus, May 23-27 (2006)
15. Krishna, A., Guan, Y., Sambatheera, C., Ghose, A. K.: Agent-based Prototyping of Web-based Systems. To appear in: The Proceedings of the 19th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA-AIE-2006), Springer-Verlag Lecture Notes in Computer Science, Annecy, France, 27-30 June (2006)
16. Koliadis, G. Vranesevic, A. Bhuiyan, M. Krishna, A. Ghose, A.: A Combined Approach for Supporting the Business Process Model Lifecycle. To appear in: The Proceedings of the 10th Pacific Asia Conference on Information Systems (PACIS), July 6-9, Kuala Lumpur, Malaysia (2006).
17. White, S. Business Process Modeling Notation (BPMN) Version 1.0. Business Process Management Initiative (BPMI.org), May (2004)
18. Giorgini, P. Kolp, M. Mylopoulos, J. Pistore, M.: The Tropos Methodology: an overview. In: Methodologies And Software Engineering For Agent Systems. Kluwer Academic Publishing (2004)
19. Fuxman, A. Liu, L. Mylopoulos, J. Pistore, M. Roveri, M. Traverso, P.: Specifying and analyzing early requirements in Tropos. In: Requirements Engineering, Springer London, 9(2) (2004) 132–150
20. Fischer, L.: Workflow Handbook 2005. Workflow Management Coalition, (WfMC) (2005)
21. Ouyang, C. W.M.P. van der Aalst, Dumas, M. and ter Hofstede, A.H.M.: Translating BPMN to BPEL. BPM Center Report BPM-06-02, BPMcenter.org, (2006)
22. Becker, J. Indulska, M. and Rosemann, M. Green, P.: Do Process Modelling Techniques Get Better? A Comparative Ontological Analysis of BPMN. In: Campbell, Bruce and Underwood, Jim and Bunker, Deborah, Eds. Proceedings 16th Australasian Conference on Information Systems, Sydney, Australia (2005)
23. Lamsweerde, A. Goal-Oriented Requirements Engineering: A Guided Tour. In: The 5th International Symp. In Requirements Engineering (RE'01), Aug. (2001)