Responding to Ongoing Change: Challenges for Information Systems Modeling

Alexei Lapouchnian, Department of Computer Science, University of Toronto, Toronto, Canada

Eric Yu, Faculty of Information and Department of Computer Science, University of Toronto, Toronto, Canada

Stephanie Deng, Faculty of Information, University of Toronto, Toronto, Canada

ABSTRACT

As modern organizations increasingly need to operate in uncertain and fast-paced business environments, pressures increase on information systems (IS) to support these enterprises in a dynamically changing world. Consequently, systems need to deliver results given incompletely known and constantly changing requirements and contexts and other uncertainties. Their development is no longer a progression from clear and stable requirements to solutions meeting them. Rather, it is a continuous process involving multiple iterations of analysis and exploration, design, and development taking into consideration changing organizational needs, available resources, and feedback from previous iterations. Since current modeling and analysis notations generally assume stable and predictable settings for IS development, this paper explores the difficulties in applying several such techniques for modeling continuously evolving systems in uncertain and rapidly changing socio-technical domains and identifies requirements for a comprehensive modeling notation suitable for these environments. Business intelligence capability implementation in enterprises is used as an illustration.

Keywords: *Adaptation, Change, Evolution, Feedback, Process Modeling, Requirements, Social Modeling*

1. INTRODUCTION

The steady improvement in communications, transportation, and international financial services and an increasing customer pressure to provide cheaper, higher-quality goods are some of the forces contributing to the global competition escalation. This puts pressure on enterprises to be highly aware of their changing business environments as well as of their own and of their competitors' performance. Changes happen fast, thus requiring companies to have strategic and operational flexibility and presenting a major challenge for information systems (IS) engineering. New, unforeseen circumstances may lead to business strategy/process reconfiguration, hence modifying IT requirements and forcing system reconfiguration or redevelopment. For enterprises to be competitive, their IT systems must not become impediments to business change. Rather, they need to actively support it by being agile and,

DOI: 10.4018/ijismd.2014100103

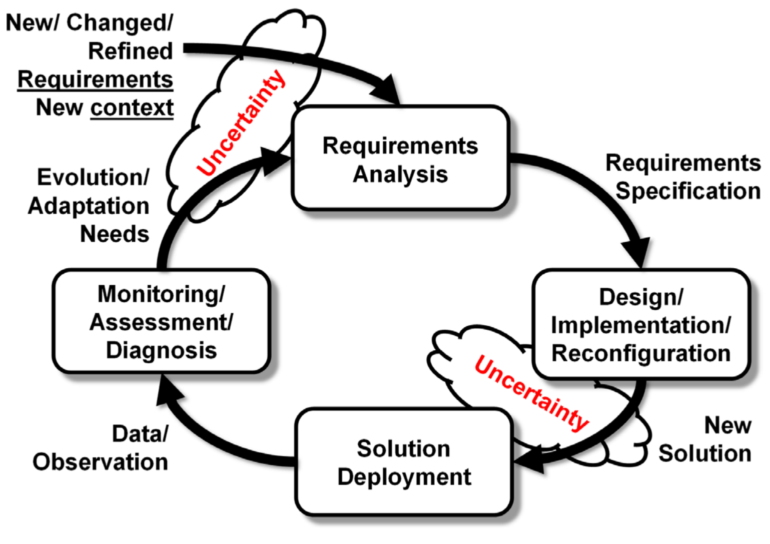
faced with the increasing amounts of raw data (from sales, sensors, social networks, network-connected devices, etc.), they need to enable its near-real-time transformation into meaningful business information.

Handling this change is a continuous process that cannot be accommodated by a single episode of a system redesign and implementation. Change needs to be monitored for and detected, the range of possible responses identified and analyzed, and the most appropriate one selected for implementation and deployment. There are numerous unknowns and uncertainties. Change occurs at many granularity levels and over many iteration cycles. Change initiatives – large and small, long-term or short-term – will encounter varying degrees of success, with lessons learned feeding back into subsequent iterations. Under these circumstances, IT systems need to continuously evolve to stay aligned with and support business-level changes. Figure 1 presents a solution lifecycle for handling such ongoing change. It applies both at business and IT levels. The figure shows two sources of uncertainty: the unpredictable nature of new/changed requirements and the possibility that solutions fail to achieve their objectives, thus requiring further system adaptation/evolution.

This presents a real challenge to traditional Requirements Engineering (RE). Given the uncertainties and the unknowns, upfront requirements analysis cannot be inadequate. Monitoring for new/changing requirements and determining if the current design meets them is vital. The challenge has prompted a call for fundamentally new approaches to requirements and new conceptual frameworks and theories for understanding them. E.g., Jarke, et al. (2011) emphasize the need for RE to support requirements during solution (and its environment) evolution at multiple abstraction levels or temporal horizons, while designs increasingly look like continuous searches for satisficing solutions. Additionally, RE needs to deal simultaneously with social and technical facets, as people and technology interact in emergent ways.

IS engineering employs modeling techniques to support requirements analysis. Process models and data models are also used extensively. Intentional and social models were introduced more recently (Mylopoulos, 1998; Nurcan, Salinesi, Souveyet, & Ralyté, 2010). Nevertheless, most current modeling techniques presume stable and predictable application settings. Can these techniques be used

Figure 1. The requirements-to-solution lifecycle


New/ Changed/ Refined Requirements New context
Uncertainty
Requirements Analysis
Requirements Specification
Evolution/ Adaptation Needs
Monitoring/ Assessment/ Diagnosis
Design/ Implementation/ Reconfiguration
Uncertainty
New Solution
Data/ Observation
Solution Deployment

to support analysis in the new, highly dynamic environments?

In this paper, we aim to: 1) motivate the need for supporting the above-described environments through appropriate modeling notations; 2) evaluate the suitability of existing social and process modeling notations for this task; 3) identify modeling requirements for a comprehensive future modeling notation designed to handle these types of systems.

To better understand the capabilities and inadequacies of existing requirements modeling techniques in highly dynamic environments, we consider the recent rapid uptake of business intelligence (BI) technologies in organizations. BI, analytics, and big data are seen as critical for advancing business performance and competitiveness (Davenport, Harris, & Morison, 2010; Manyika, et al., 2011). They are perceived as capable of closing the feedback loop to realize the vision of a self-adaptive enterprise. The socio-technical trajectory of how BI took shape in organizations provides a microcosm of the challenges of uncertain and evolving requirements. The trajectory reflects the increasingly common situation where people and technology have to adapt to each other. Multiple adaptation rounds may occur, as initial responses are often inadequate. Furthermore, adaptations frequently introduce technological innovations for reducing change response times. These dynamic and adaptive phenomena in today's enterprise IT environments present new challenges for modeling and analysis.

The article is structured as follows. We first look at the previous work on adaptive systems modeling and identify their inadequacies for the above-outlined challenges. We then apply two current notations to model a typical business-driven BI (BDBI) adoption path in enterprises. As each stage of implementation unfolds, we examine how well the goal-based social modeling technique *i** (Yu, 2009; Yu, Giorgini, Maiden, & Mylopoulos, 2011) supports the analysis of the relevant issues. Its limitations are noted. In attempting to model multiple layers of change processes, we employ the widely-known BPMN (Object Management Group, 2011) notation. Again, we note its limitations and introduce improvised extensions for our purpose. We then summarize the encountered modeling issues and outline a set of requirements for a comprehensive modeling framework for conceptualizing requirements in an increasingly dynamic world and conclude the paper.

This paper extends Yu, Lapouchnian, and Deng (2013) by refining the motivation, adding details to the case study, enhancing the investigation of the two notations' applicability to modeling ongoing change, and improving the analysis of requirements for a future comprehensive modeling notation for that purpose.

2. RELATED WORK

Adaptive enterprises can gain insights into their environment and internal operations and quickly react to changes, threats, and opportunities. Handling change is a major concern of Enterprise Architecture (EA). EA has been advancing both as a field of study and in practice. Enterprise modeling benefits are becoming widely recognized, as evidenced by the progress in standards, e.g., The Open Group (2012). However, current EA frameworks and languages do not support explicit modeling or analysis methods for dealing with change and adaptation. Current enterprise models focus mainly on expressing a single architectural state of the enterprise (e.g., the as-is state or some to-be state). Today's methods, e.g., the ADM in TOGAF (The Open Group, 2011), provide guidance on large-scale architectural transformations, from an as-is to a to-be state, but do not address the full range of enterprise dynamics, e.g., frequent, shorter-timeframe instance-level adaptations. Business processes (BPs), while robust and well-established, do not offer much variability or support for inter-process relationships and other dynamic aspects that are relevant in dynamically changing domains. Current models are therefore too static and restrictive for architecting today's enterprises functioning in fast-moving and uncertain settings. Still, proposals exist for adaptive EAs. Wilkinson (2006) offers a method for designing an adaptive EA by utilizing the adaptability of a service-oriented computing infrastructure, while Hoogervorst (2004) recommends integrating business and IT views to support agility and change. Also, while several studies looked at the role of IT in business agility, e.g., Mathiassen and Pries-Heje (2006), they have not considered the mutual iterative adaptation of business and IT.

Enterprise adaptability today cannot be achieved without the supporting IT systems being similarly adaptive. While software maintenance has long focused on modifying and evolving systems (e.g., at the source code level (Madhavji, Fernandez-Ramil, & Perry, 2006)), recently, the emphasis shifted to the architecture-based adaptation (Oreizy, et al., 1999) owing to software architectures' support for modularity, layering, modifiability, etc. Due to modern business complexity and dynamism, there is a growing interest in self-adaptive systems (SAS) that can adapt to changes in their environments, recover from failures, etc. – all without human intervention (Cheng, et al., 2009).

System dynamics and control systems theory possess a wealth of well-established techniques for modeling and analyzing dynamic systems (Meadows and Wright, 2009). Feedback ideas from control systems were incorporated into software engineering (SE) (e.g., MAPE loops (Kephart & Chess, 2003)), where they are extensively used in SAS (Cheng, et al., 2009), and into business management (e.g., SIDA loops (Haeckel, 1999)). These approaches feature a clear separation between the target system and the adaptation mechanism. The prevalence of data analytics today promotes the feedback loop introduction into enterprises, although they may not be explicitly viewed as such.

RE research has long focused on requirements change, e.g., (Rolland, Salinesi, & Etien, 2004; Nurmuliani, Zowghi, & Powell, 2004). Software evolution has also been a major preoccupation in SE, e.g., (Madhavji, et al., 2006). These studies mostly concentrated on SE artifacts and processes and not on the organizational environment's dynamics. They have not considered mutual adaptation or co-evolution between business users and IT from a socio-technical perspective. While also addressing adaptations occurring within software systems, recent approaches focused on the importance of requirements in SAS, including the need to treat requirements as runtime entities (Sawyer, Bencomo, Whittle, Letier, & Finkelstein, 2010) and monitor their attainment (Souza, Lapouchnian, Robinson, & Mylopoulos, 2011), the need for novel requirements types for adaptation, such as the awareness requirements leading to feedback loops in SAS (Souza, et al., 2011) and the requirements capturing possible requirements evolution trajectories (Souza, Lapouchnian, Angelopoulos, & Mylopoulos, 2013). These ideas will likely prove useful in adaptive enterprises.

Iterative optimization of organizational processes is the key feature at the highest level of maturity in the CMMI maturity model (SEI, 2010). Agile methods in software development also aim for high adaptivity. However, they do not encompass adaptation in the user environment.

Overall, the research described above does not offer modeling and analysis approaches that would be suitable for socio-technical domains characterized by dynamically and unpredictably changing requirements, contexts, etc. necessitating continuous design evolution informed by context monitoring and supported by feedback loops and multiple processes with complex interactions. In this paper, upon applying two existing notations to an example scenario, we highlight the resulting difficulties and identify a number of requirements modeling notations need to satisfy if they are to be successfully applied in these domains.

3. MODELING THE ONGOING EVOLUTION OF BUSINESS-DRIVEN BI

We now study the suitability of two existing modeling techniques to a real-life example exhibiting many characteristics discussed above and identify their limitations.

To examine the challenge of handling ongoing change, we consider the current rapid spread of BI technologies. Despite the data analytics' much touted business benefits, BI adoption has not been straightforward. The needs and requirements of business users are uncertain and evolve in hard-to-predict ways. High expectations and business pressures create unprecedented demands on IT departments. Industry observers have noted several rounds of innovation and adaptation in enterprise BI (Eckerson, 2012a; Eckerson, 2012b). For our example, we draw on Eckerson's analysis of BDBI (or self-service BI) (Eckerson, 2012b) based on a survey of 234 BI practitioners from business and IT across multiple industries. Our industry contacts reported similar experiences. Overall, BDBI is an ongoing battle that is widely discussed by industry observers and analysts (e.g., Forrester (Evelson, 2012)).

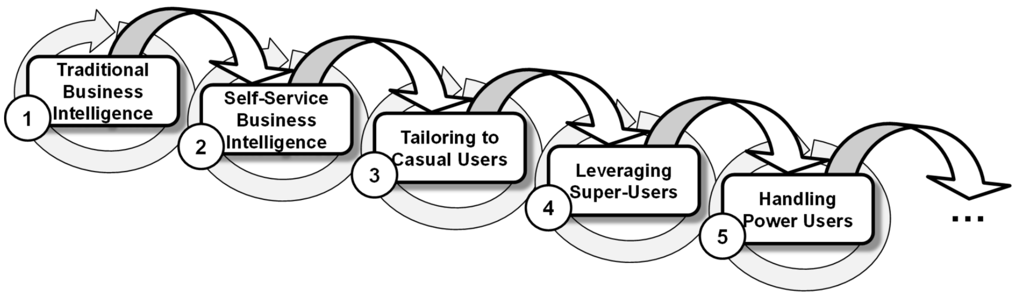
In the following subsections, we examine a series of socio-technical configurations in the evolution of business-driven BI – a trajectory of BI adoption that was typical to many companies. We focus on the implementation of BI reporting and not on the BI infrastructure development since the former lets us emphasize the co-evolution of social and technical aspects of solutions while omitting many BI implementation details for better accessibility. At each stage in the evolution, we concentrate on the pressures of change and how they prompted a transition to a new configuration. Figure 2 illustrates the five BI configurations discussed in this paper. Once deployed, each solution is monitored (circular arrows represent feedback loops), with its deficiencies analyzed, and then evolved into the next solution. Further evolutions were left out of this paper's scope. Note that we cannot compare these solutions as alternatives to select the best one since each of them applies in a particular state of the evolving domain and addresses requirements specific to that state.

We attempt at using models to reveal pertinent issues and to support analysis. From amongst existing modeling techniques, we have elected to use *i**, a modeling notation created to help analyze information systems in social contexts. With each evolution stage involving both technical and organizational changes, we expect a social modeling technique like *i** to offer suitable support for analysis. We use business-driven BI to uncover the limitations of this technique when applied to highly dynamic socio-technical settings. We assume the context to be a dynamic business domain with rapid changes in analytical needs.

3.1. Stage 1: Traditional BI

BI technologies are designed to facilitate organizational decision-making. A *traditional BI* environment is usually implemented with data from transactional databases. The data are collected, cleansed and massaged through ETL (Extract, Transform, and Load) processes, stored in a corporate Data Warehouse (DW), and used to produce business reports utilized by managers to make operational or planning decisions.

Figure 2. Presented BI adoption solutions


1
Traditional Business Intelligence
2
Self-Service Business Intelligence
3
Tailoring to Casual Users
4
Leveraging Super-Users
5
Handling Power Users
...

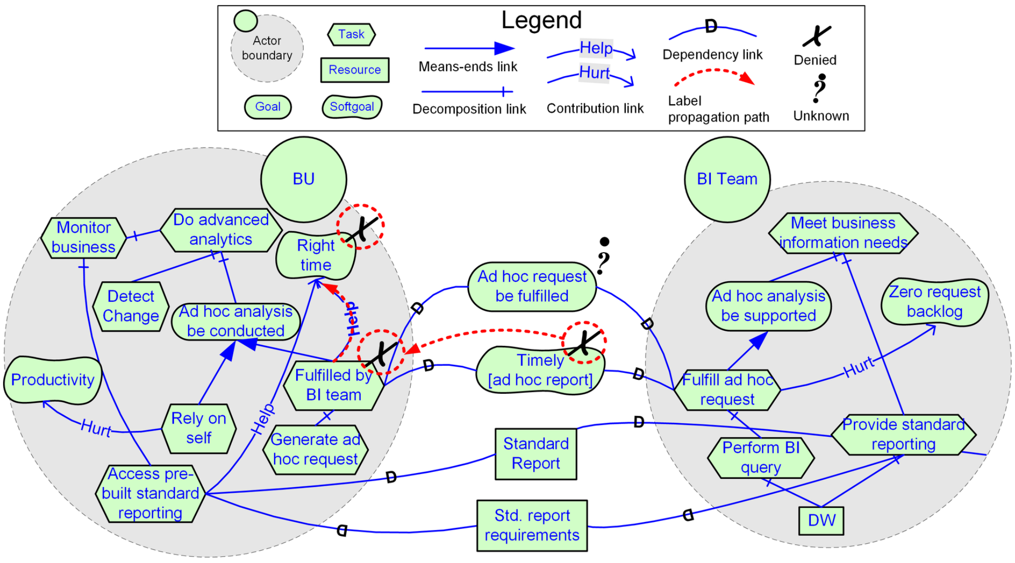
To address the information needs of Business Users (BUs), the corporate BI Team develops standard reports for monitoring KPIs and to answer pre-determined business questions. BUs can generate these reports themselves as they deliver just the right information effortlessly to non-technical users while drastically lowering the ongoing workload on BI Team. However, BUs often need additional (ad hoc) reports and expect BI Team to deliver them promptly. Here, BUs' lack of technical skills was judged as an important barrier to change, so BI Team was tasked with evolving the enterprise data analysis capability, while the BUs' responsibility was to identify new data analysis needs. The initial report set covered most of BUs' needs and the scheme where BI Team occasionally produced new reports when requested by BUs was deemed flexible enough. The expectation was that the analysis requirements change rate would not be high and the new report design cycle would generally match it. However, in a dynamic domain with explosive data growth, BI Team struggled to deliver timely ad hoc reports. The gap between the system's rigid data analysis capabilities and the BUs' changing analytical needs was increasing. The time cycle for the sole option of reducing this gap was significantly slower than the high rate of hard-to-anticipate business needs changes. Ultimately, BUs' needs were not met in this configuration.

Given this failure, could appropriate models have helped anticipate it, understand what happened, and possibly resolve it by finding a better solution? To answer these questions, we try using *i**, a goal-oriented requirement modeling approach focusing on actors' mutual dependencies for goals to be achieved, tasks to be performed, and resources to be furnished. Within each actor, means-ends links from tasks to goals can be used to answer "why" questions. Multiple tasks for achieving the same goal represent alternative choices available to that actor. Softgoals (quality goals) serve as criteria for choosing among alternatives. The notion of a softgoal in *i** draws upon the treatment of non-functional requirements (NFRs) in SE (Chung, Nixon, Mylopoulos, & Yu, 2000). Softgoals can also feature in dependencies.

In traditional BI (Figure 3), BU depends on BI Team for a *Standard report*. To *Provide standard reporting,* BI Team depends on *Standard report requirements* from BU. (*italic* font is used to refer to objects in models.) In BU, *Access pre-built standard reporting* is a sub-task in *Monitor business*. For business questions not covered by standard reporting, BU utilizes the task *Do advanced analytics*. It is further refined into the task *Detect change [in business environment]*, in which BU looks for changes in the external business environment, and the goal *Ad hoc analysis be conducted*. This goal can be *Fulfilled by BI Team* (right means-ends branch in BU), which in turn depends on BI Team to achieve the goal *Ad hoc request be fulfilled*. Also, *Ad hoc report* needs to be *Timely* – a softgoal dependency. As described earlier, this softgoal is not met by BI Team (marked by *X*), leading to failed goals in BU (note the label propagation path in Figure 3, which is shown for illustrative purposes only – in practical *i** modeling it is not used) and affecting BU's ability to make timely decisions (*Right time*).

A social, goal modeling approach as exemplified by *i** supports goal achievement analysis in actors and factors that contribute to or impede it. Figure 3 shows that in traditional BI, the long-established "design precedes access" (Eckerson, 2012a) method works well for structured/repeated analysis (e.g., monthly sales reports, profitability and financial analysis). However, as business becomes faster-paced and uncertainties increase, decision makers cannot predetermine what questions need answering and when. They increasingly rely on ad hoc analysis for decision-making under unanticipated conditions. Challenges for BI Team are exacerbated as data volumes increase, data sources multiply, and new types of analytics are introduced. This produces an insurmountable backlog for BI Team. Frustrated users are then tempted to bypass BI Team by choosing the alternative *Rely on self* to achieve *Ad hoc analysis be conducted*, but that would hurt their *Productivity*. Thus, no alternative modeled in Figure 3 can meet all the BUs' goals. To address the unmet timely ad hoc reports goal, BI Team could look for other ways to achieve *Ad hoc*

Figure 3. Failure of the traditional BI setting modeled using i (Adapted From Yu, et al., 2013)*



analysis be supported. Indeed, it creates a new option in the next solution iteration.

The *i** model partially helps in answering our previous questions. To facilitate understanding of what happened, goal-based models can reveal the driving forces for change. The task *Detect change* captures who is responsible for monitoring the business environment (where change originates) and how monitoring relates to ad hoc reporting. Further, the model identifies unmet goals and, through label propagation, their impact on the organization. However, the *i** model of Figure 3 is a snapshot that fails to capture the scenario dynamics. Speed and change rates cannot be expressed. Thus, the bottleneck resulting from BI Team's inability to handle report requests can only be indirectly inferred. Frequencies of occurrences are unclear. E.g., the *Standard report requirements* dependency is infrequent, whereas the *Ad hoc report* dependency is increasingly frequent. Therefore, the model could not have predicted the failure. However, goal-based models' strength is in capturing and analyzing goal achievement alternatives: Figure 3 shows two ways of doing ad hoc analysis. Unfortunately, both are impractical. We look at whether and how process models can help capture and analyze system/organizational dynamics later in the paper.

3.2. Stage 2: Moving to Self-Service BI

Since the excessive reliance on BI Team is a major reason for the initial solution's failure, some organizations adopted self-service BI tools to meet BUs' ad hoc needs in a timely way. Companies embraced such tools because they support users' direct access to business data without IT departments being the intermediaries. They attempt to empower BUs and shorten the ad hoc report design cycle by moving it from BI Team to BUs. Business users have to rely on themselves, so BI Team can be freed from the stream of ad hoc requests. Moreover, the solution appears to accommodate different types of business users, including Power Users (PUs) – statisticians, data scientists, etc. – who are data analysis experts tasked with uncovering business insights from raw data through creative and iterative ad hoc analyses, which, if proven useful, are integrated into the standard report portfolio. Enterprises use these insights to gain competitive advantage over rivals. Here, BUs still monitor for business environment changes.

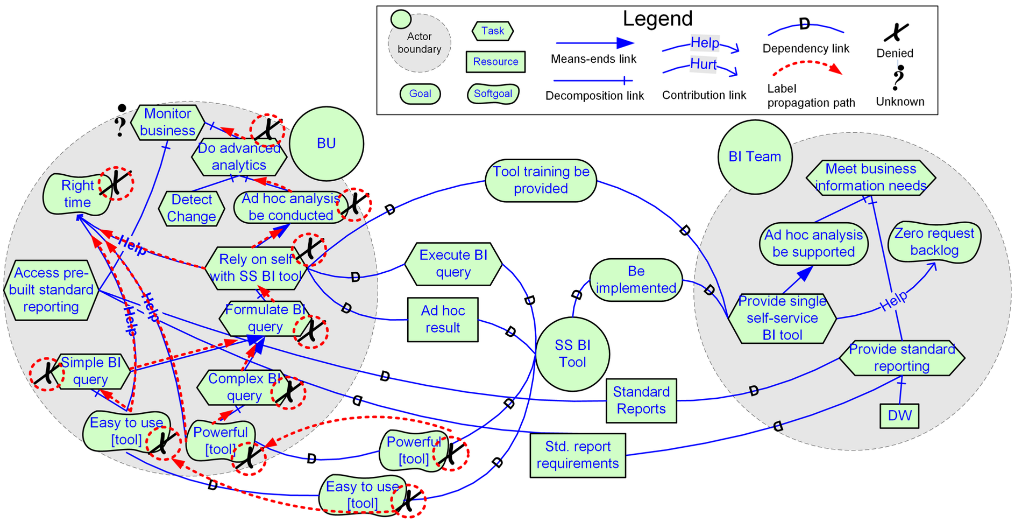
While modeling this solution iteration, we want to capture the need for the tool to be both easy and powerful and the shift of responsibility for report creation to BUs themselves and how it affects their objectives, including timeliness. Is it possible to show the shorter report design cycle?

In Figure 4, BI Team decides to *Provide Single Self-Service BI Tool* to help achieve its *Ad Hoc Analysis be Supported* goal. This new option is hoped to result in *Zero Request Backlog* (for ad hoc reports). With the self-service tool, what was handled by BI Team (*Formulate and Execute BI Query*) is now done by BUs. Since BUs do the ad hoc analysis themselves when needed, the obstacle to *Right Time* is overcome by removing the dependency on others.

This solution assumes the users have the skills to utilize self-service tools. However, this one-size-fits-all solution was generally unsuccessful. While providing more flexibility than the initial approach, self-service BI did not fully accommodate the needs of both user classes who need to perform different query types and therefore require different level of freedom (in terms of reports, tools, and data access). Casual Users (CUs), including executives, managers and frontline workers, are domain experts who use information to do their job. They primarily employ standard reports, but increasingly need ad hoc analysis for fast decisions. The solution intimidated these BUs by demanding too much skill from them, while simultaneously hindering the performance of PUs, who have strong analytical aptitude, good tool proficiency, reasonable domain knowledge, and prefer unfettered raw data access. While BU needs are more clearly understood, PUs' needs are rather unpredictable since they do data experimentation and any constraints imposed by analysis tools impede their work. Thus, the tool flexibility was too much for one user category and too little for the other, and the complexity and inflexibility of the tools were the main barriers to the self-service BI success. This illustrates the need for modeling, analysis, and continuous monitoring of user needs and capabilities.

In Figure 4, we model the above challenge as failures of the two competing tool-related softgoals, *Easy to Use* and *Powerful*. Starting from the failed identically named dependencies (in *i**, dependencies are labelled with

Figure 4. Social model capturing the failure of self-service BI (Adapted From Yu et al., 2013)



dependums, around which dependency relationships center) from BU to Self-Service BI Tool, the softgoal failures are propagated (see the dashed propagation path) through the model of BU causing failures in both *Simple BI query* and *Complex BI query* and thus preventing BUs from formulating queries and, in turn, not allowing them to *Rely on self with Self-Service BI tool*. With no ability to rely on IT for ad hoc reporting, this failure inevitably leads to the failure of BU's advanced analytics task.

The above problem points to different user categories possibly having different usability requirements. Not accounting for users with diverse objectives and capabilities/skills in the design and in the solution models leads to the system failure. Splitting the user base into categories is a design decision about granularity. Doing it upfront or delaying the split until later may yield different results in terms of addressing various usability requirements. Therefore, the modeling capability to represent and reason about design granularity is important. This includes capturing goals, criteria, and choices that differentiate user groups, product types, etc., possibly leading to separate designs for them.

In terms of modeling, we see that *i** can somewhat show that the ad hoc report design is now "closer" to BUs as they are responsible for queries, though the dynamic behaviour is still hard to analyze. Cyclical behaviours and differences between cycles cannot be shown. User capabilities/skills have not been explicitly modeled in this iteration, but we attempt this next.

3.3. Stage 3: Tailoring to the Casual User

This approach recognizes the different needs of BU classes and provides solutions fit to specific user groups. Here, the self-service tool is tailored to CUs for ad hoc report creation. Thus, the driving force behind this change is the need to address a particular user subset. Focusing on CUs, we produce a model reflecting their point of view. What are the assumptions that we make about CUs when designing the self-service tool for them? What do CUs want from the tool? Can the design satisfy them?

The tool is intended for CUs to be *Quick to Learn* and *Easy to Use* when compiling their own ad hoc reports (see Figure 5). BI Team provides plenty of tool training to CUs upon implementing the tool (see the dependency *Tool Training be provided* and the BI Team task fulfilling it) expecting that CUs will *Learn tool*, thus enabling them to formulate and execute BI queries by themselves.

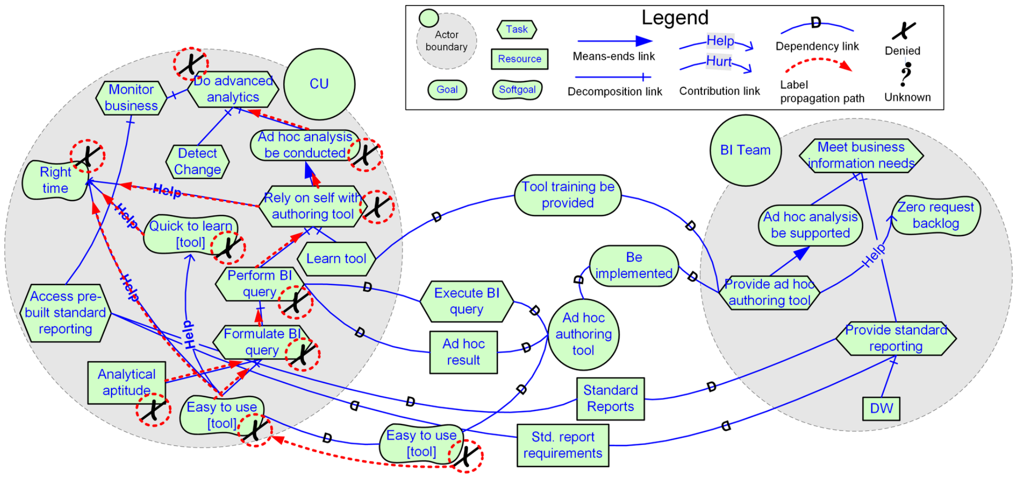
Unfortunately, tool adoption proves complicated. Figure 5 helps visualize the tool adoption obstacles inside CU. CUs must *Learn Tool* before they can formulate and perform BI queries. While learning is infrequent (a one-time effort per feature), the learning curve is insurmountable for many CUs. Even with ample training, tool learning is harder than expected (the softgoal *Quick to learn* fails in CU), especially since users do not consider it *Easy to Use*. The longer the learning, the slower the users adopt new tool features. This hinders getting quick answers (*Right time*) for frequently changing business questions. Also, without proper motivation and incentives, CUs might lack tool learning effort.

Furthermore, to *Formulate BI Query*, one needs to be analytically inclined. While tool knowledge can be acquired and tool skills trained, many CUs lack *Analytical Aptitude,* which requires a longer time to develop (the resource *Analytical Aptitude* fails). This is often the reason why CUs cannot run queries themselves. The CUs' lack of *Analytical Aptitude* points to the overly optimistic assumptions about their skills embedded in this design.

*i** distinguishes among physical actors (agents) and logical actors (roles). A capability is a property of an agent. Agents may fail to possess or acquire the skills required for playing logical roles. *Analytical Aptitude* illustrates that we model human capabilities as resources within actor boundaries. Failure to obtain such resource indicates the unavailability of the corresponding capability.

In this iteration, certain *i** features – the ability to model requirements from a user

Figure 5. Failure to tailor to needs of casual users: Analysis using a social model (Adapted From Yu et al., 2013)



group's point of view, to represent both roles and concrete agents, and to capture the required user capabilities – proved extremely useful. Still, *i** cannot model the gradual learning and user skill improvement and the evolving tool functionality to match those skills. The label propagation path in Figure 5 shows that failures in this BI solution critically affect CU's objectives. If modeled in real-life BI implementation projects, the discovery of unrealistic assumptions about user capabilities could have triggered analysis/discussion, likely leading to the conclusion about the proposed solution's infeasibility.

3.4. Stage 4: Leveraging the Super-User

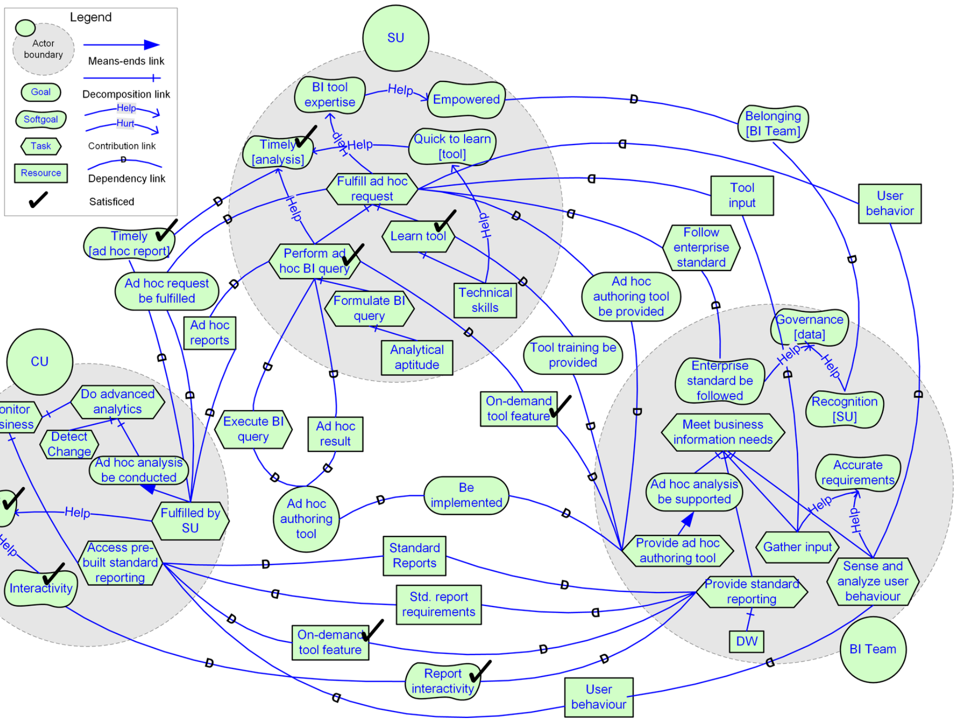
Super-Users (SUs) are leveraged here to alleviate the tool adoption problem afflicting the previous solution. A SU is a technically proficient CU that has become the departmental go-to-person for tailored information. BI Team provides the ad hoc report authoring tool to SUs rather than CUs while giving CUs interactive standard reporting tools, which allow some view personalization. A configurable tool enabling on-demand feature provisioning (for both CUs and SUs) to deal with uncertain and evolving user capabilities and demands is utilized.

The model produced for this iteration should help verify that with appropriate assumptions about their skills, SUs can actually provide timely ad hoc reports to CUs. Moreover, we want to analyze if SUs, being BUs themselves, are motivated to help CUs besides fulfilling their own objectives and if the sense of belonging to BI Team can be fostered in them. Similarly, the model should help determine if this 3-tier BI organization meets its objectives. Finally, can the multiple report design and tool improvement cycles be made to work in the new approach?

As shown in Figure 6, SUs have the report authoring tool, so CUs now depend on SUs for timely ad hoc reports and the report design cycle was moved to a SU. A SU represents a social component added to the technical solution. Unlike the previous approaches, the new intermediary separates the user and the data. The dramatic reconfiguration of actor relationships is clearly visible in Figure 6.

This social landscape change helps overcome the CUs' lack of personal interest and tool skills, which caused problems previously. Compared to CUs, SUs are tech-savvy and analytically inclined (see the *Technical Skills* and *Analytical Aptitude* resources in SU, respectively), while gravitating towards using BI tools. SUs quickly become proficient with the

Figure 6. Introducing a super-user: A big social landscape change modeled in i (Adapted From Yu et al., 2013)*



report authoring tool. Moreover, by fulfilling ad hoc requests for CUs in their departments, SUs become known and respected for their *Tool Expertise*, which makes them feel *Empowered*. CUs are given the interactive standard reporting (*Standard Reports* and *Report Interactivity* dependencies between CU and BI Team) with personalizable views.

The *On-demand Tool Feature* dependency is fulfilled by BI Team for both user categories. It offers the flexibility to deal with changes in user capabilities and needs. The sensing mechanism is implemented through actor dependencies. To better understand user needs (*Accurate Requirements* in BI Team), BI Team not only *Gathers Input* from users, but also *Senses and Analyzes User Needs*. This allows tool features to be adjusted gradually, without overwhelming or limiting users.

This approach has multiple levels of iterative design or adaptation cycles to achieve the required responsiveness (*Right time* in CU). CUs can quickly and easily tailor analysis given their changing information needs through interactive standard reports (e.g., predefined drill-down, personalization of colours and fonts). This achieves timeliness through a nearly instantaneous adaptation cycle. Changes beyond simple customizations are deferred to SUs, who have the tool/skills to support a fast adaptation cycle for per-user changes. BI Team publishes new standard reports regularly as requirements continue to evolve.

While this approach seemingly solves the prominent issues that emerged in the analytical environment, there are potential alignment problems that require careful handling. How does one ensure that personal interests of a user are aligned with SU's objectives when that user is assigned the role? How do we properly align SUs to the interests of BI Team (i.e., have them *Follow the Enterprise Standard*)? Incentives are needed to prevent misalignment between physical agents and the roles they are playing, and between different actors in the BI landscape. E.g., some BI managers can recognize the SUs' value and incorporate them into BI Team (the *Belonging* dependency), thus making them feel *Empowered* (in SU) and therefore better aligning their work with corporate standards.

Therefore, as expected, the model in Figure 6 can capture this solution's dramatic social landscape change – the lighter load on CUs and the heavy involvement of SUs. It also supports the analysis of whether SUs' personal goals (e.g., *Empowered*) are met and how. Similarly, SUs' skills and their effect on SUs' objectives and the solution's success are explicitly represented. However, while some design cycle elements can be modeled (e.g., sensing as resource dependencies), the cycles' relative length and other properties are hard to capture and analyze.

3.5. Stage 5: Catering to and Reigning in the Power Users

While lacking space to present models for further solution iterations, here we discuss ongoing developments and challenges in BDBI. The initial PU-tailored solution addresses the problem of the one-size-fits-all BI tool overconstraining PUs' analytical work. Replacing the self-service tool, an analytical toolbox is given to PUs for creative data exploration, enabling PUs to select the tools they find suitable for their analysis. Although PUs now have the tool flexibility, two challenging issues remain. For comprehensive data analysis, a PU also needs unconstrained data access. The current ways to access data – sending queries to BI Team or creating PU's own offline data sets – limit the possibility for iterative analysis and create data maintenance issues. The isolated data shadow systems jeopardize data consistency and make data governance difficult. Also, due to limited computing resources on desktops and local servers, PUs often experience poor system performance on their complex queries.

The refined approach fully addressing PUs' needs is to deploy analytical sandboxes. Sandboxes are powerful environments for efficient data manipulation. The sandbox solution satisfies both PUs and BI Team. It makes PUs first-class citizens in the corporate BI environment. The sandbox's powerful computation capability dramatically improves query performance. Data accessibility is simplified through online data access and the ability to upload and merge local data. Offline data set maintenance difficulties are also alleviated. Subsequently, it helps BI Team achieve data governance and improve data consistency. Additionally, the centralized analytical environment helps promote collaboration and reuse between PUs.

The BDBI setting is highly dynamic and yet still emerging. To unfold and resolve the complexities illustrated above, organizations need experimentation with different BI solutions. They are iterative adaptation processes with co-evolution of technical and human systems. Solutions are not limited to the illustrated approaches, as uncertainty and ongoing change still reign. E.g., is the new role, the SU, well aligned with the interests of others? How does one support PUs' creative data exploration when a complex analytical job runs differently every time and the data needed to answer unexpected questions are not yet collected?

4. MODELING TEMPORAL AND ITERATIVE CONCEPTS

As discussed above, *i** models cannot represent temporal change (though they can help reason about the motives and forces driving change) and consequently adaptation loops and design cycles. Here, we look at multiple layers of change in dynamic environments and, while noting its limitations, attempt to use the extended BPMN notation to model multiple change processes.

4.1. Multiple Levels of Adaptation Processes

Using *i** models, one cannot recognize that changes are taking place within different time frames or scales. E.g., activities monitoring business performance and leading to management actions occur at different time frames than the monitored transactional activities. Tool (re) design to support the BI activity occurs at yet another (less frequent) time frame. To depict and better understand these processes and their relationships we look to process modeling.

The process model (Figure 7) shows processes operating within their own time cycles. Sales transactions (*Operational Process*) can be started every few seconds/minutes. When monitoring sales performance (*Business Monitoring*), a BU determines the tracked KPI values within minutes by looking at dashboards. However, it may take hours to understand why certain sales declined or what products are likely to be the most profitable in the next quarter. While we attempted to employ the widely adopted BPMN notation to convey the relationships among different kinds of change processes, we encountered the need for adaptations and extensions. Pools, which can represent abstract roles in BPMN, are used to show a multiplicity of relatively independently operating processes. Each process has a typical duration giving a rough sense of their relative timing, another extension of the BPMN notation. Process durations are important for analysis. E.g., if an operational process runs in seconds (we may have thousands of instances of those) and the monitoring/change processes take days, the latter will not be possible to adapt operational processes on a per-instance basis.

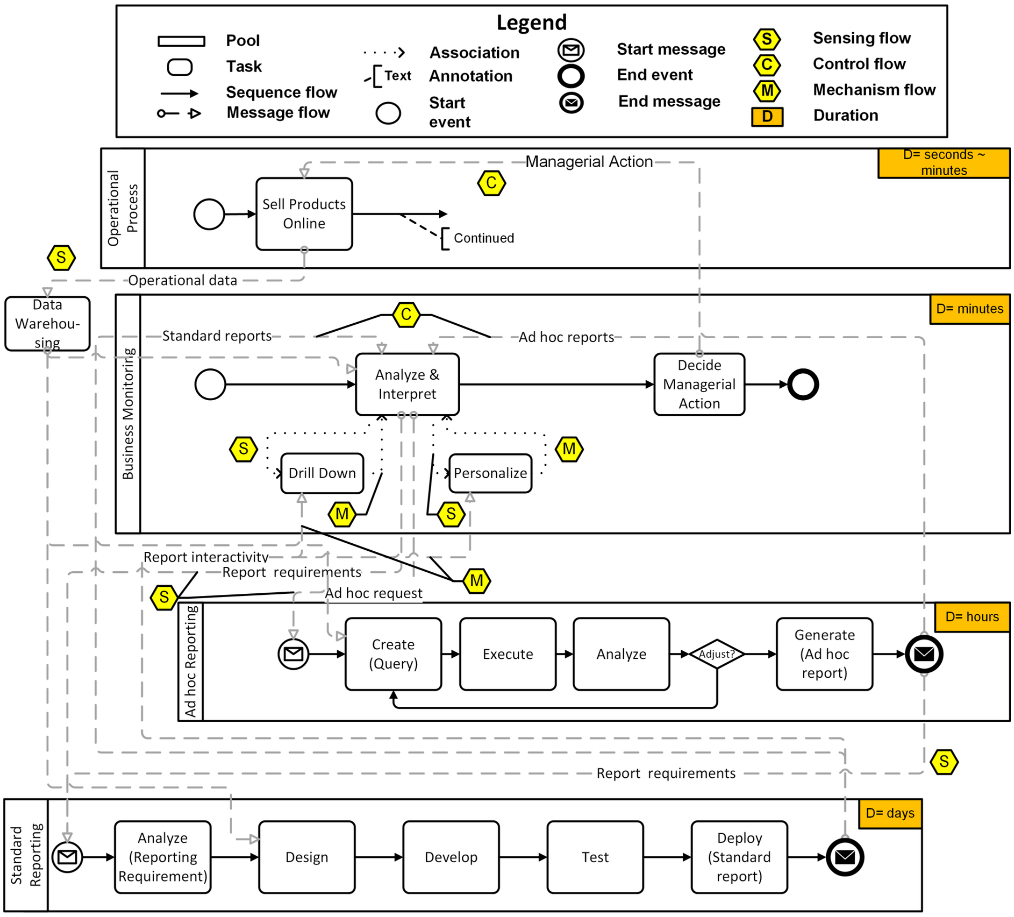
We are particularly interested in the types of inter-process connections rather than sequence flows within processes. How do these processes interact with each other? What is the nature of the inter-process flows? From the BI adoption example, note that *Business Monitoring* is a change process with respect to *Operational Process*. The output data from the *Sell Products Online* activity is a sensing operation extracting information (via BI infrastructure such as a data warehouse) about that activity to improve it. This information is compiled into reports for use by *Business Monitoring*. The change process results in the *Managerial Action* altering how *Sell Products Online* operates. For enterprise adaptiveness, an analyst needs to identify such monitoring/change process for *Sell Products Online* to evaluate this feedback loop's effectiveness regarding timeliness, frequencies, effectiveness of the managerial action, etc.

The *Managerial Action* flow is not a normal transactional data input transformed by every instance of the *Sell Products Online* process. Rather, it is a "control" input prescribing or modifying how the *Sell Products Online* process operates. It varies infrequently compared to the transaction frequency of the operational process.

To interpret business performance properly, BUs drill down or personalize reports as they see fit (sensing operation). (We note the non-standard use of BPMN here.) The outputs of these activities do not directly "control" the process (*Analyze and Interpret*) they aim to improve. Rather, they produce better tools enabling or supporting their target processes. The tool here helps tailor the information by changing its scope or format. *Report Interactivity* capability enables these features. We call this type of flow a "mechanism" flow.

More BMPN annotations are used here to differentiate the newly identified flows from the normal inputs/outputs. Sensing operations (e.g., *Operational Data*) are message flows annotated with <*S*>, showing as outputs from the bottom of activities. Control flows (e.g., *Managerial Action*) are message flows labelled with <*C*>, showing as inputs to the top of activities. And the <*M*> annotation represents mechanisms (inputs to the bottom of activities, e.g. *Report Interactivity*). The "control" and "mechanism" flow terminology is borrowed from the IDEF0 language (NIST, 1993). In this paper, BPMN was selected over IDEF0 due to it being a newer, more widely used and a more flexible notation. Besides annotating different inputs and outputs, we expand the use of "message" flows between pools to represent artifacts (e.g., requirements, reports) and capabilities (e.g., tools).

Figure 7. Modeling sensing, control, and mechanism flows in multiple design cycles (Adapted from Yu et al., 2013)



Report Requirements and *Ad Hoc Request* are annotated as sensing outputs as they provide inputs to change processes (*Standard Reporting* and *Ad Hoc Reporting* respectively), which will deliver new reports for the next use cycle. Reports help investigate available information (upon some processing), not the raw data. Hence, we consider reports (*Standard Reports* and *Ad Hoc Reports*) as control inputs that constrain users' information analysis capability.

4.2. Adaptation Loops and Feedback

By differentiating control and mechanism inputs from normal inputs and sensing from normal outputs we can locate adaptive loops, which are important for modeling and analysis of adaptive systems/enterprises since these loops identify and handle change. The concern is not the repetitive execution of the same activities, but the existence of the information flow back to the next iteration. First, we identify the <*S*>-<*C*> (Figure 8A) and <*S*>-<*M*> flow pairs. Then, for each pair, we can identify the activities comprising the corresponding feedback loop. E.g., Figure 8B shows the MAPE feedback loop conceptualization (Kephart & Chess, 2003) controlling the operational process. Here, *Sell Products Online* corresponds to the MAPE loop's "execute" phase, *Decide Managerial Action* corresponds to "plan", etc.

Adaptive loops reveal special relationships between processes. A higher-level process is a design (change) process with respect to its lower-level processes (the "use" or "execution" process). It senses how well its lower-level process executes and may change the operation of the latter. Change is enacted in two ways – through a "control" flow, where a higher-level process adapts the target process by selecting/constraining the possible options for it at run-time, or a "mechanism" flow, which changes the space of options for its target process by creating new capabilities, thus evolving the target process (Souza, et al., 2013).

There is a hierarchy among processes reflecting their relative control order (see Figure 9). We draw horizontal boundaries between processes operating at different design levels. The BPMN collapsed process element represents a "collapsed" pool with its internal details hidden, allowing us to focus on inter-process relationships.

Managerial actions from *Business Monitoring* improve *Operational Process*. *Standard Reporting* provides reports that support *Business Monitoring* and help answer known/anticipated questions. When this basic analysis is insufficient, *Ad Hoc Reporting* is triggered addressing additional information needs. However, ad hoc reports can only leverage existing data. Hence, if a business question requires new data for analysis, *Data Exploration* is triggered to meet the need. While *Tool Development* builds the analytical infrastructure and tools (leveraged by the other processes), *Feature Provisioning* provides on-demand features based on user capabilities and needs.

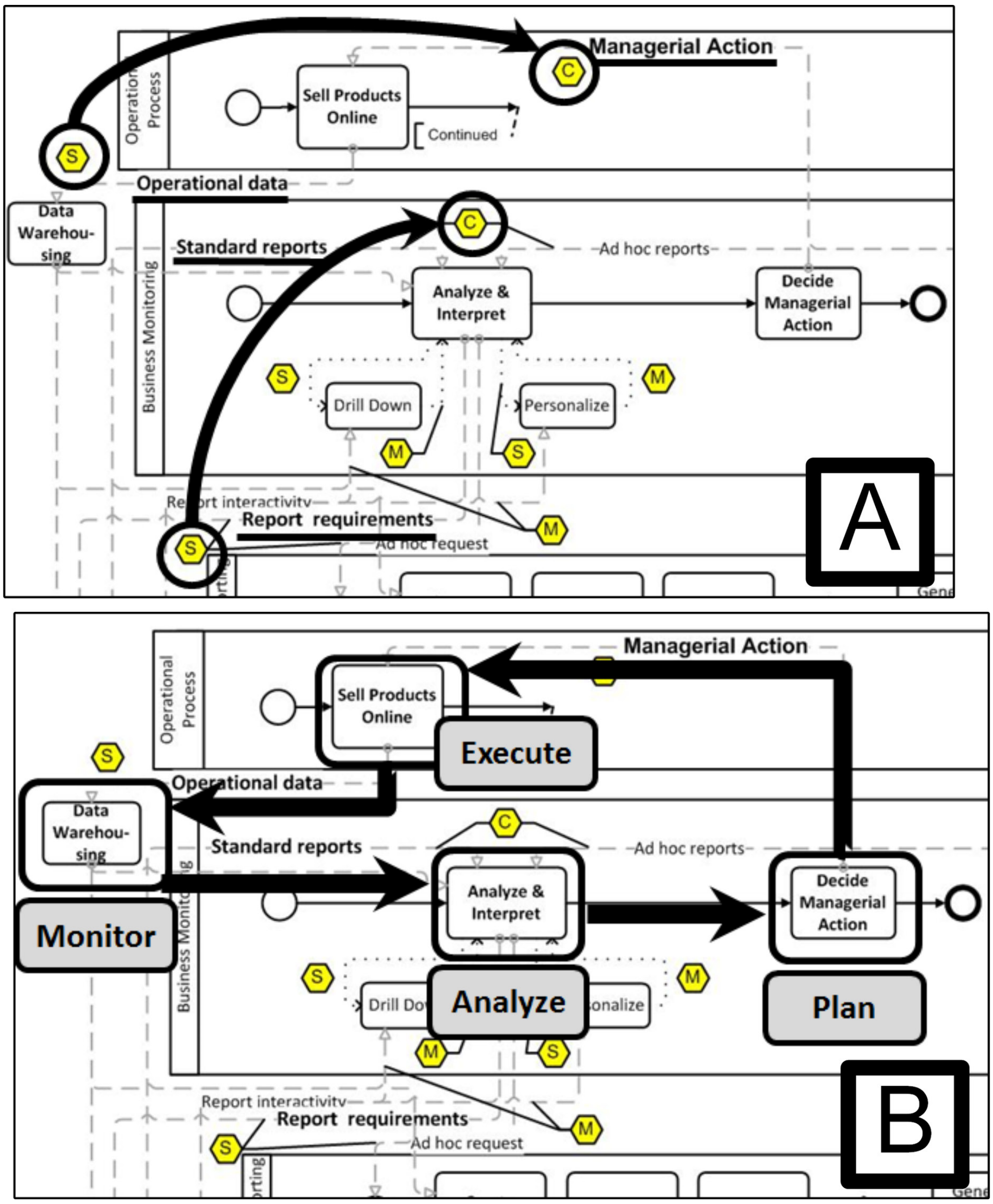
To identify these loop patterns, we adopted a directional convention for the inputs/outputs similar to IDEF0 (as shown in the legend of Figure 9). At the top right, processes *B* (*Business Monitoring*) and *A* (*Operational Process*) form a feedback loop, as previously discussed. *B* is at a higher level as it is a design process for *A* (boundary 1). Moving downward, both processes *D* (*Standard Reporting*) and *C* (*Ad Hoc Reporting*) are report design processes for *B*. Therefore, a level boundary (boundary 2) is placed between processes *C/D* and *B*, meaning that *B* is a lower-level process w.r.t. *C* and *D*. Yet, process *E* (*Data Exploration*) sources new data and evolves the semantic layer for *C*. So, *E* is a higher-level process w.r.t. *C* (boundary 3). At the bottom left, another boundary (boundary 4) exists between process *F* and those above it. Processes *F* (*Feature Provisioning*) and *G* (*Tool Development*) produce analytical tools for several processes above. The two operate at different time frames, as *G* runs on a much longer cycle (months) than *F* (hours to days). Similarly, *C* (hours) delivers more frequent and faster results than *D* (days).

We introduced levels in Figure 9 to help recognize the inter-process design relationships. The need for a process design activity distinct from process execution arises when a process change cannot be readily accomplished at "runtime", forcing a redesign. However, if runtime adaptation capabilities (certain options and/or configurations) are built-in, adaptation can occur within the same level, without another design cycle, thus achieving better agility and responsiveness. For example, a CU can drill down and personalize predefined reports (Figure 7), which is a self-adaptive loop helping tailor information analysis at runtime.

5. DISCUSSION

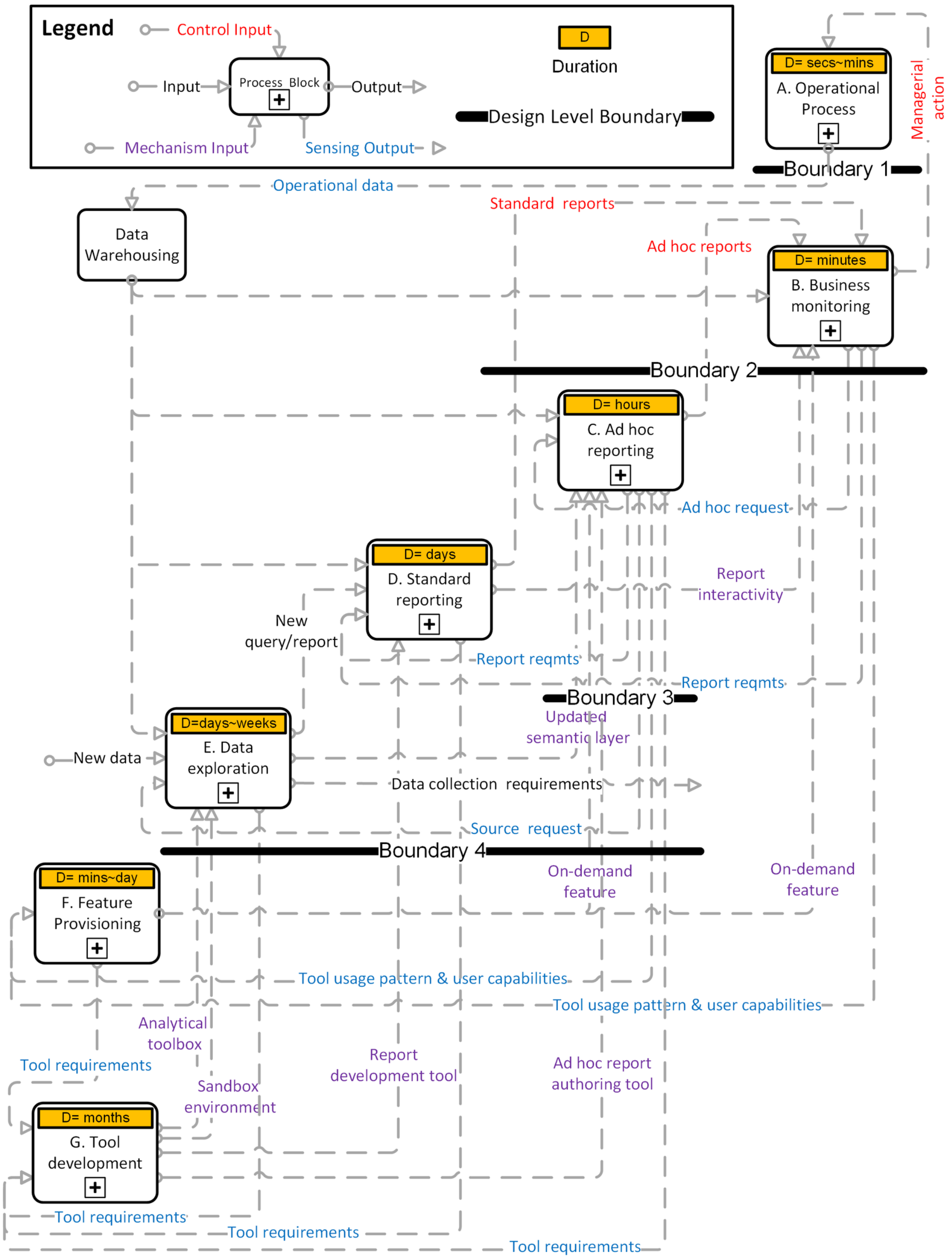
In the previous section, we tried applying social and process modeling notations to capture and analyze a problem facing many enterprises today – the creation of a BI solution that is adaptive, resilient, effective, efficient, timely, flexible, user friendly and empowering for all the involved roles. That was an example of a system characterized by continuous and unpredictable requirements evolution, iterative development, and the need for socio-technical solutions. We tested the (extended) modeling capabilities of these notations to determine if their combination could adequately capture the complexities of the BI scenario. Through multiple solution iterations we showed that while certain important domain/solution

Figure 8. Using process models to identify sense-control pairs (A) and a MAPE loop (B)



aspects could be modeled, others could not. We stress that we do not promote the presented models as constituting a solid approach for modeling these systems. While we believe that a combination of social and process notations could be a reasonable starting point for a comprehensive modeling/analysis approach, its full development remains future work. That notation should be able, e.g., to help answer some of the questions we discussed earlier (solution failure anticipation, reasons behind failures and alternatives identification). Below

Figure 9. Using an extended BPMN model to represent design level boundaries (Adapted from Yu et al., 2013)



we identify important modeling and analysis aspects, which this future notation would need to address, together with a discussion based on our BI example experience on how *i** and extended BPMN fared with respect to these modeling requirements.

5.1. Combining Modeling Notations

When combining several modeling notations, one faces the problems of integrating them, of their consistency, traceability, etc. Our philosophy is requirements-driven, and we believe in starting at the intentional level, with social and goal-oriented *i** models that support the identification and modeling of business needs, relevant stakeholders, and the overall social landscape of enterprises. These models inform and motivate process design within organizations. However, completely greenfield projects are rare, especially in the context of the continuous solution evolution we described in this paper. Therefore, there needs to be flexibility in which order the notations are used.

In our example, BPMN and *i** model things from different perspectives and levels of abstraction, thus complementing each other. Therefore, one-to-one correspondence between every element of the two models is impossible (e.g., some process-level details may be too low-level for *i**). One research effort tackled a similar problem in the context of reconfigurable BPs (Lapouchnian, Yu, & Mylopoulos, 2007). The authors proposed a semi-automatic generation of process models from goal models that preserved the goal-level variability at process level and maintained some traceability and consistency among models. Adopting a similar approach would support the traceability from most *i** elements to process model elements, but the backward traceability would not be guaranteed as process models are much more detailed.

5.2. Variability Modeling and Binding

Support for modeling and analysis of intentional variability (how system objectives can be met), the criteria for selecting among the available choices, and the behavioural variability implementation is important for designing flexible, robust, adaptable/adaptive organizations and systems. *i** is capable of means-ends goal analysis, while explicitly representing the corresponding selection criteria. BPMN models can capture varying behaviour and flows, but cannot represent quality criteria for choosing appropriate configurations.

Further influencing or limiting alternatives selection are constraints and barriers – requirements regarding resources, skills, capabilities, production/computing capacity, etc. associated with variants. Analysis of these barriers is thus important as they influence enterprise flexibility/adaptivity and can trigger processes aimed at overcoming the barriers (e.g., learning the reporting tool). We found that certain barriers can be modeled as resources in *i** (e.g., the SU's analytical aptitude in Figure 6), but a more thorough representation of these constraints in *i**, in process models, or using another notation must be explored.

5.3. Social Modeling

Business enterprises are social systems and therefore to support flexibility under changing and unknown requirements, one needs to consider socio-technical solutions to enterprise objectives. *i** is a social modeling notation having extensive capabilities to capture and analyze social aspects of enterprises. These include means-ends analysis done from individual actors' point of view, with actor-specific quality criteria (NFRs) guiding the selection of suitable alternatives, exploration of alternative dependency configurations and responsibility assignments (e.g., the ad hoc report development responsibility), and the ability to distinguish logical and physical actors and thus to determine if individual agents' skills match the requirements of the roles they have to play – otherwise this becomes a change barrier. Also, social alignment issues – conflicting actor goals and conflicts involving agents' personal goals and the objectives of the roles they are playing – can be analyzed. In our example, making SUs feel empowered in their positions is an attempt to align their individual interests with those of the enterprise. However, temporal aspects, such as the gradual increase in agents' skill level, are hard to model in *i**.

While requirements engineering approaches in SE and business process management are traditionally focused on abstract actors, requirements methods prevalent in human computer interaction, user-centered design, etc. concentrate on agents. To deal with changing and evolving requirements in socio-technical systems we need to integrate both perspectives.

5.4. Multiple Levels of Design

One important lesson learned from the BI example is that in dynamic domains with unpredictably evolving requirements design decisions should remain open for revisions and adding new solutions to existing ones should be possible (e.g., *i** supports this via the openness of its means-ends decomposition). Thus, the overall space of configurations includes not only the ones currently implemented and enabled at runtime, but also those resulting from some level of enterprise redesign. As shown in our example, upon a failure of one solution, a significant enterprise redesign usually ensues, with new tools and/or actors being introduced.

In a hierarchy of design levels, a particular level uses the design provided by a higher-up design level and makes design decisions for a lower design level. Different design levels usually operate on different time cycles (e.g., the frequent ad hoc report design by SUs vs. the rare on-demand tool feature provisioning by BI Team, see Figure 6). Moreover, design levels have different decision-making criteria that *i** models can capture. Organizational roles responsible for deliberations at different design levels are likely also different. E.g., higher design levels are usually responsible for strategic decisions, which are therefore delegated to upper management.

When handling change, adaptation can happen both within the same design level (through switching to an already implemented alternative) or across multiple levels. Several situations can result in the latter. First, the variability within the original design level might not be sufficient to successfully handle a change (e.g., when no standard report can support a new data analysis requirement, thus needing an ad hoc report). Then, upper design levels will evolve the original level by providing new ways of achieving its objectives. This evolution can amount to a large development project. Second, the need for upper levels to handle change may be due to organizational inflexibilities where certain decisions need the approval of high-level managers. As illustrated above, adaptations involving multiple design levels are likely caused by serious change barriers.

5.5. Feedback

An enterprise operating in a dynamic, unpredictably evolving domain needs to adapt to the changing environment (context), evolving requirements, or failures to achieve its objectives (or utilize opportunities for their improvement). This requires sense-and-response behaviour exemplified by feedback loops consisting of controllers and target systems. The controller monitors its target system and can modify it if the monitored output deviates from the expected value. A target system modification by its controller represents adaptation. Thus, feedback loops feed information to controllers to improve their target systems' next iteration cycle. As already discussed, we can capture feedback loop details with <*M*>, <*S*>, and <*C*> flows (where control flow roughly corresponds to adaptation, while mechanism flow amounts to evolution) in annotated process models. Feedback loops can operate within a single design level, with that level exemplifying a self-adaptive system, or across multiple design levels.

An important aspect of feedback affecting adaptation cycle timing is how the need for change is determined. In the simplest case, a control error in the target process will result in a change applied to its next iteration. This makes the adaptation fast, and the adaptation cycle will generally match the target process cycle. However, this adaptation type frequently leads to an undesirable oscillating behaviour (repeated target overshooting and undershooting). To alleviate this problem, one needs to consider not only the present error (proportional control), but also the accumulation of past errors (integral control) and/or the error's rate of change (derivative control). This requires studying multiple target process instances before making adaptation decisions, which slows the adaptation rate and illustrates that monitoring decisions affect adaptation cycles.

To analyze enterprise adaptiveness we need to map the paths of various (often nested) adaptive loops criss-crossing the organization to determine whether they achieve the desired adaptive behaviour. Positive and negative feedback should be analyzed. Dynamic systems analysis techniques may be relevant (e.g., time-domain response, change rates and frequencies). Since an adaptive enterprise will encounter transformational and disruptive change, the modeling framework will need to encompass nonlinearities. A proposal in Souza, et al. (2011) identifies awareness requirements (requirements about the success/failure of other requirements, as in "*Ad Hoc Analysis be Conducted* should never fail") as those that feedback loops are designed to achieve. Capturing these requirements can potentially help integrating multiple, possibly conflicting feedback loops at the intentional level. As illustrated in Souza, et al. (2011), *i** needs appropriate annotations to support these requirements.

Looking at the BI solutions described in this paper, one can recognize a missing high-level feedback loop aimed at achieving adequate (including timely) support for the BUs' evolving business analytics needs and for improving BI-driven decision making through identifying new data dependencies, etc. (i.e., the activities of power users). While this loop is not explicitly modeled, the analysis of every solution iteration presented above – including the identification of the preceding solution's failure, the reasons for that failure, and the possible alternative solutions – happens within it. The BI adoption trajectory demonstrates several of its iterations.

5.6. Modeling Temporal and Dynamic Aspects

Since *i** has poor support for quantities, frequencies and temporal progression, one cannot distinguish between failures on a per-instance transactional level and inherently invalidated options. Modified process models analyzed in this paper can capture process duration and frequency, the rate of change and the corresponding design cycle length, leading to the possibility of bottleneck identification and improved coordination. To avoid overloading the models, we have not distinguished between control/adaptation applied to a particular process instance or to all process instances henceforth. For detailed analysis of the temporal/real-time properties of various solutions, specialized (e.g., queuing) models can be utilized. Unlike *i**, process models, with their support for events and messages, can capture change triggers and specify change propagation within enterprises. Causal loop diagrams from system dynamics can also augment the analysis where appropriate. They can help entertain what-if scenarios, while stock and flow diagrams can model dynamically changing resources or skill levels. Still, intentional models are invaluable to represent variability in handling change and the rationale for selecting alternatives.

6. CONCLUSION

As the pace of change accelerates and complexity increases, one needs novel modeling techniques to support requirements activities in the new setting. Our vision of a socio-technical system as an artifact being continuously redesigned by the accompanying processes is supported by the ideas of Garud et al. (2009), who argue that in the environments characterized by continuous change, requirements are not well-defined, preferences are fluid, and solutions "emerge in action". In this context, design goals remain a constantly moving target, which necessitates a continuous (re)design process.

In the BI adoption example, we found that co-evolution of the social and the technical can be modeled and analyzed *to some degree* using *i** modeling. Overall, the *i** notation proved suitable for modeling and analyzing social aspects of solutions, including actors, their objectives, skills assumptions, and dependencies. Given the models capturing such interdependencies, reallocation of responsibility across actor boundaries (including automation boundaries) from one solution to the next can be modeled and analyzed, with results feeding into subsequent solutions. Failed functional and non-functional system and actor goals, which drive change, can also be modeled, as can unrealistic expectations about agents' capabilities. However, *i** models are snapshots and cannot express solution dynamics or convey cyclical, recurring patterns of handling change. To complement the *i** models, we tried using extended process modeling to depict multiple design levels. Several modeling challenges and issues were uncovered as a result of the exercise, suggesting the need for a comprehensive rethinking of requirements modeling techniques in the new context. These new techniques will be the basis for a model-based approach to adaptive enterprise architecture (Yu, Deng & Sasmal, 2012).

The modeling challenges identified in the BI example are not unique. In recent and ongoing developments, like the spread of web-based IS, enterprise 2.0 and social media, enterprise search, etc., uncertain and evolving requirements and the uncertainty of whether solutions will achieve those requirements in complex and changing environments are pressing challenges to be answered by the IS engineering community.

Our study uses scenarios from a single domain setting and applies only two modeling techniques, which are limitations. Currently, we are working towards a comprehensive notation to address the challenges and fill in the gaps outlined in this paper, which will incorporate social, process, and other types of modeling. We are looking at a number of diverse application areas to help both the notation development and its validation. We will also aim at tackling the practical modeling issues, such as model complexity and scalability.

ACKNOWLEDGMENT

This research was supported by the NSERC (Natural Sciences and Engineering Research Council of Canada) Business Intelligence Network.

REFERENCES

Cheng, B., de Lemos, R., Giese, H., Inverardi, P., & Magee, J. et al. (2009). Software engineering for self-adaptive systems: A research roadmap. *Software Engineering for Self-Adaptive Systems* [Berlin Heidelberg: Springer-Verlag.]. *LNCS*, *5525*, 1–26.

Chung, L., Nixon, B., Mylopoulos, J., & Yu, E. (2000). *Non-Functional Requirements in Software Engineering*. Norwell, MA: Kluwer. doi:10.1007/978-1-4615-5269-7

Davenport, T., Harris, J., & Morison, R. (2010). *Analytics at work: smarter decisions, better results*. Boston, MA: Harvard Business Press.

Eckerson, W. (2012a). *Big data and its impact on data warehousing*. BeyeResearch/TechTarget. Retrieved October 26, 2013, from http://research.b-eye-network.com/data/document.do?res_id=1336149536_429

Eckerson, W. (2012b). *Business-driven BI: using new technologies to foster self-service access to insights*. BeyeResearch/TechTarget. Retrieved October 26, 2013, from http://searchcio.rl.techtarget.com.au/detail/RES/1346794557_442.html

Evelson, B. (2012). *The Forrester wave: self-service business intelligence platforms, Q2, 2012*. Forrester Research.

Garud, R., Jain, S., & Tuertscher, P. (2009). Incomplete by Design and Designing for Incompleteness. In K. Lyytinen, P. Loucopoulos, J. Mylopoulos, & W. Robinson (Eds.), *Design Requirements Engineering: A Ten-Year Perspective* (pp. 137–156). Berlin, Heidelberg: Springer. doi:10.1007/978-3-540-92966-6_7

Haeckel, S. H. (1999). *Adaptive Enterprise: Creating and Leading Sense-And-Respond Organizations*. Boston, MA: Harvard Business Press.

Hoogervorst, J. A. P. (2004). Enterprise Architecture: Enabling Integration, Agility and Change. *Int. J. Cooperative Inf. Syst.*, *13*(3), 213–233. doi:10.1142/S021884300400095X

Jarke, M., Loucopoulos, P., Lyytinen, K., Mylopoulos, J., & Robinson, W. (2011). The brave new world of design requirements. *Information Systems*, *36*(7), 992–1008. doi:10.1016/j.is.2011.04.003

Kephart, J., & Chess, D. (2003). The vision of autonomic computing. *IEEE Computer*, *36*(1), 41–50. doi:10.1109/MC.2003.1160055

Lapouchnian, A., Yu, Y., & Mylopoulos, J. (2007). Requirements-Driven Design and Configuration Management of Business Processes. *International Conference on Business Process Management (BPM 2007)*, Brisbane, Australia, Sep 24-28, 2007. doi:10.1007/978-3-540-75183-0_18

Madhavji, N., Fernandez-Ramil, J., & Perry, D. (2006). *Software evolution and feedback: theory and practice*. Hoboken, NJ: John Wiley & Sons. doi:10.1002/0470871822

Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., & Hung Byers, A. (2011). *Big data: the next frontier for innovation, competition, and productivity*. McKinsey Global Institute.

Mathiassen, L., & Pries-Heje, J. (2006). Business agility and diffusion of information technology. *European Journal of Information Systems*, *15*(2), 116–119. doi:10.1057/palgrave.ejis.3000610

Meadows, D. H., & Wright, D. (2009). *Thinking in Systems: A Primer*. White River Junction, VT: Chelsea Green Publishing.

Mylopoulos, J. (1998). Information modeling in the time of the revolution. *Information Systems*, *23*(3-4), 127–155. doi:10.1016/S0306-4379(98)00005-2

NIST. (1993). Integration Definition for Function Modeling (IDEF0). Retrieved October 26, 2013 from http://www.idef.com/pdf/idef0.pdf

Nurcan, S., Salinesi, C., Souveyet, C., & Ralyté, J. (Eds.). (2010). *Intentional perspectives on information systems engineering*. Berlin, Heidelberg: Springer. doi:10.1007/978-3-642-12544-7

Nurmuliani, N., Zowghi, D., & Powell, S. (2004). Analysis of requirements volatility during software development life cycle. *Software Engineering Conference*. doi:10.1109/ASWEC.2004.1290455

Object Management Group. (2011). *Business Process Model and Notation (BPMN), Version 2.0*. Retrieved October 26, 2013, from http://www.omg.org/spec/BPMN/2.0/PDF/

Open Group, The. (2011). *TOGAF Version 9.1*. Retrieved October 26, 2013 from, http://pubs.opengroup.org/architecture/togaf9-doc/arch/

Open Group, The. (2012). *ArchiMate 2.0 Specification*. Retrieved October 26, 2013, from http://pubs.opengroup.org/architecture/archimate2-doc/

Oreizy, P., Gorlick, M., Taylor, R., Heimbigner, D., Johnson, G., & Medvidovic, N. et al. (1999). An Architecture-Based Approach to Self-Adaptive Software. *IEEE Intelligent Systems*, *14*(3), 54–62. doi:10.1109/5254.769885

Rolland, C., Salinesi, C., & Etien, A. (2004). Eliciting gaps in requirements change. *Requirements Engineering*, *9*(1), 1–15. doi:10.1007/s00766-003-0168-y

Sawyer, P., Bencomo, N., Whittle, J., Letier, E., & Finkelstein, A. (2010). Requirements-aware systems: a research agenda for re for self-adaptive systems. *International Requirements Engineering Conference*, pp. 95–103. doi:10.1109/RE.2010.21

Software Engineering Institute (SEI). (2010). CMMI Version 1.3 Information Center. Retrieved October 26, 2013, from http://cmmiinstitute.com/cmmi-solutions/cmmi-version-1-3-information-center/

Souza, V., Lapouchnian, A., Angelopoulos, K., & Mylopoulos, J. (2013). Requirements-driven software evolution. *Computer Science -. Research for Development*, *28*(4), 311–329.

Souza, V., Lapouchnian, A., Robinson, W., & Mylopoulos, J. (2011). Awareness requirements for adaptive systems. *International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2011)*, Hawaii, USA.

Wilkinson, M. (2006). Designing an 'Adaptive' Enterprise Architecture. *BT Technology Journal*, *24*(4), 81–92. doi:10.1007/s10550-006-0099-5

Yu, E. (2009). Social Modeling and i*. In Borgida et al. (Eds.), *Conceptual Modeling - Foundations and Applications*. Berlin, Heidelberg: Springer. doi:10.1007/978-3-642-02463-4_7

Yu, E., Deng, S., & Sasmal, D. (2012). Enterprise architecture for the adaptive enterprise – a vision paper. *Workshop on Trends in Enterprise Architecture Research (TEAR 2012)*, Open Group Conference, Barcelona, Spain. doi:10.1007/978-3-642-34163-2_9

Yu, E., Giorgini, P., Maiden, N., & Mylopoulos, J. (2011). *Social Modeling for Requirements Engineering*. Cambridge, MA: MIT Press.

Yu, E., Lapouchnian, A., & Deng, S. (2013). Adapting to Uncertain and Evolving Enterprise Requirements: The case of business-driven business intelligence. *International Conference on Research Challenges in Information Science (RCIS 2013)*, Paris, France, May 29-31, 2013. doi:10.1109/RCIS.2013.6577687

Alexei Lapouchnian obtained his Ph.D. from the University of Toronto, Canada and was subsequently a postdoctoral fellow at the University of Toronto and the University of Trento, Italy. He is interested in requirements-driven approaches for exploiting goal and contextual variability for modeling, analysis, and design of customizable, evolving, and adaptive software and socio-technical systems, business processes and enterprise architectures. His previous work included the use of high-level requirements models at runtime for system reconfiguration and adaptation as well as the identification and analysis of new types of requirements (e.g., awareness requirements) that lead to self-adaptive functionality in systems. He is currently a business analyst and a researcher at the University of Toronto.

Eric S. K. Yu is Professor at the University of Toronto, Canada. His research interests are in the areas of information systems modeling and design, requirements engineering, knowledge management, and software engineering. Books he has co-authored or co-edited include: Social Modeling for Requirements Engineering (MIT Press, 2011); Conceptual Modeling: Foundations and Applications (Springer, 2009); and Non-Functional Requirements in Software Engineering (Springer, 2000). He is an associate editor for the Int. Journal of Information Systems Modeling and Design, and serves on the editorial boards of the Int. J. of Agent Oriented Software Engineering, IET Software, and the Journal of Data Semantics. He is co-editor for the MIT Press book series on Information Systems. He was Program Co-chair for the Int. Conference on Conceptual Modeling (ER'08, ER'14).

Stephanie Deng obtained her Master's degree from the iSchool at the University of Toronto, Canada. Upon graduation, she joined the research project "BI-enabled Adaptive Enterprise Architecture". She is interested in applying social modeling and analysis in real business setting. She also has interests in the areas of Enterprise Architecture, Business Intelligence, and Business Process Management. With over 10 years' experience in managing information systems, she is now an Application Architect at the Heart and Stroke Foundation Canada.