Analyzing the Evolution of Software Development: from Creative Chaos to Software Ecosystems

Mahsa Hasani Sadi
Department of Computer Science
University of Toronto
Toronto, Canada
mhsadi@cs.toronto.edu

Eric Yu
Faculty of Information
University of Toronto
Toronto, Canada
eric.yu@utoronto.ca

***Abstract*— As a software organization matures and expands, it often evolves through different styles of organization, for example, beginning with creative chaos as a start-up, then introducing disciplined processes to raise quality, and later regaining agility through light-weight practices. Recently, many firms join collaborative networks to develop software products and platforms for a shared market, constituting "Software Ecosystems". At each stage of evolution, the software organization aims to overcome critical challenges faced in its earlier stages, while balancing business, organizational, social, and technical forces of change. To illustrate how the evolutionary trajectory of a software development firm is shaped by various interacting forces, we draw upon a longitudinal case study taken from the literature. We use the *i** strategic actors modeling framework to help analyze the forces that trigger the transition from one organizational configuration to another.**

Keywords—Software Development Organization; Evolution; Software Ecosystems; Agile Development; Product Line Engineering; Waterfall; Creative Chaos; Modeling & Analysis; Strategic Actors.

I. INTRODUCTION

As a software development firm grows, its needs, objectives and environment change correspondingly. To survive in the face of these internal and external changes, the organization must adapt its processes, structures and behavior on an ongoing basis. As a result, a software development enterprise often experiences various evolutionary stages during its life cycle; for instance, beginning in Creative Chaos as a start-up, then adopting disciplined processes to control growth and raise quality, and later turning to light-weight Agile development practices to balance between discipline in production and flexibility in responding to changing customer demands [1]. Recently, due to economic and business incentives, the software development industry is shifting towards Software Ecosystems (SECO) in which firms join collaborative networks to develop software products and platforms for a shared market [2, 3, 4, 5].

As a software organization moves from one stage to another, it has to overcome the challenges of previous stages, to align its objectives and organizational configurations with the new environment, and to seek balance amongst various business, organizational, technical, and social forces of change.

As noted in a recent study analyzing the state of the art in software development research and practice [6], current engineering methods and techniques that deal with evolution and adaptation focus on changes that occur in software products and in software development processes. They overlook the changes that are also occurring at the same time in organizational, social, and business configurations. Furthermore, current approaches tend to attend to one software product or product line at a time, in the context of a single project within one software development organization. Increasingly however, change needs to be understood and analyzed on a broader temporal and spatial span, involving many business and social actors over longer periods of time.

In this paper, we aim to identify the challenges arising from the broader perspective of evolution where business, organizational, and social forces interact and co-evolve with technical products and processes in software development, and to obtain the requirements for suitable analysis and modeling techniques. To this objective, we examine the evolutionary trajectory of a software development organization over many years, drawing upon a longitudinal study from the literature [7]. We use the *i** strategic actors modeling framework [8], which has received wide experimentation for modeling sociotechnical settings, to uncover and analyze the pertinent issues.

The rest of this paper is organized as follows: In section 2, the related literature is reviewed and the contributions of this research are highlighted. In section 3, we analyze the evolutionary trajectory of a software development organization. As each stage of the organization unfolds, we uncover and analyze the driving forces of change using the *i** modeling framework. The implications from modeling and analysis, and the challenging issues are discussed in section 4. Finally, we summarize the outcomes of this research and conclude in section 5.

II. RELATED RESEARCH

In this section, we review the related literature and highlight the contributions of this research. The research efforts related to this work can be categorized in three classes:

1) Software Evolution. Extensive research has been devoted to the analysis of software evolution, including [9, 10, 11, 12, 13, 14, 15]. In this class of research efforts, evolution is viewed as a software design problem and the main focus is on

theorizing and addressing the evolution of software artifacts and processes. Correspondingly, methods and systematic tools have been developed to support and manage the evolution of software products throughout their lifecycle.

However, in this research, we adopt a novel perspective by analyzing the evolution of software development in the context of a software development firm. In the context of an enterprise, the evolution of software development is intertwined with various social, organizational and business forces, which are as influential as technical concerns in shaping the evolutionary trajectory of software development activities. To the best of our knowledge, the only research effort close to this research is [16].

In [16], the evolutionary development of a single software system has been analyzed in the socio-technical environment of an enterprise, using the *i** framework. Therein, it has been argued that the evolution of software systems is in fact a co-evolution triggered by various changing organizational and social forces surrounding the software system. In contrast, in this research, we adopt a higher level of abstraction and analyze the evolution of software development activities in an organization. Using *i** modeling, we illustrate that the evolutionary trajectory of software production is shaped by the co-development of various evolving business, organizational and social aspects in an enterprise. We also demonstrate that the scope of co-evolution may transcend the boundaries of an organization and include its external environment.

2) Software Adaptation. Adaptation of software systems and artifacts with the changing needs and situation of the environment is the focus of self-adaptive software systems [17, 18]. In this class of research, adaptation and reconfiguration is mainly discussed as aligning the structure and behavior of a single software system with its environment. Adaptation has also been discussed at the level of software development activities and processes in Agile software development [19]. Agile development methods accommodate adaptive mechanisms in the software process models to align software products with the changing demands of customers. The scope of adaptation in Agile processes is bounded to the re-configuration of software development activities, products and teams within a single development project.

However, herein, we argue that adaptation is also a crucial concern for managing the evolutionary development of a software development organization. Moreover, we illustrate the ongoing reconfiguration of a software development firm in response to its changing needs and environment over time. At an enterprise level, various business, organizational, social, and technical issues are introduced in the adaptation process which transcend the temporal and spatial boundaries of a single software project and a single product. In this research, we model and analyze how co-adaptation occurs between organizational configurations, business needs, software development activities, and social aspects, in a software development firm.

3) Modeling Development and Evolution of Software. Numerous models and meta-models have been proposed to analyze different aspects of software development. For example, Software Process Engineering Meta-model 2.0 (SPEM 2.0) [20] adopts a process-centered view towards the analysis of software development activities, and focuses on the software development activities, roles, and products. Software Ecosystems Meta-model (SEM) [21] models business and technical networks of relationships among several software development firms. However, each of these meta-models focuses on one specific aspect of software development. To analyze the evolution of software development, a modeling and analysis technique is required which can capture and relate the heterogeneous elements of the context.

Moreover, specific modeling and visualization techniques have been proposed to analyze the evolution and adaptation of different aspects of software development activities: HISMO meta-model for representing the history and time dimension in the evolution of software artifacts in [22]; Technical Ecosystems Modeling Notation (TECMO) meta-model [23] for analyzing the evolution of software artifacts in Software Ecosystems; and graph models in [24] to analyze the evolution of software components over time. However, the proposed modeling approaches mainly focus on the evolution of system aspects in software development.

Additionally, to understand and visualize the evolution of environment and social aspects in software development, statistical models and biological models have been used in [25] and [26] respectively. Causal models have also been applied to address the co-evolution of organizational, business, social, and technical factors in software projects and software processes [27, 28]. In this class of efforts, to address the heterogeneity of environmental elements involved in software evolution, models with quantitative assumptions in the background are used. However, since available data on the evolution of software development mostly exists in the form of descriptive texts and reports, we adopt a qualitative reasoning approach to analyze the evolution of software development, which is intrinsically more consistent with the context of available information.

III. Analyzing the Evolution of a Software Development Organziation

To help understand and analyze what various forces of change are involved in the evolution of software development activities and how they drive evolution, in this section, we use the *i** modeling framework [8]. *i** is a modeling approach, with the main focus on how, in a network of strategic actors, actors depend on each other, in order to fulfill their goals, to perform their tasks and to furnish their resources. Within the boundaries of each actor, means-ends links from tasks to goals can be used to answer "Why" questions behind actors strategic decisions. Multiple tasks for achieving the same goal represent variability in means selection. The notion of the soft goal in *i** can be used to explicate the quality criteria to select from different means or identify qualitative goals in actors' dependencies. Given characteristics such as the heterogeneity of involved factors in the evolution of a software development organization, *i** might be expected to offer suitable support for analyzing and reasoning about the evolution of software development. We examine how the capabilities of *i** can be utilized to understand and analyze the evolutionary trajectories of software development firms.

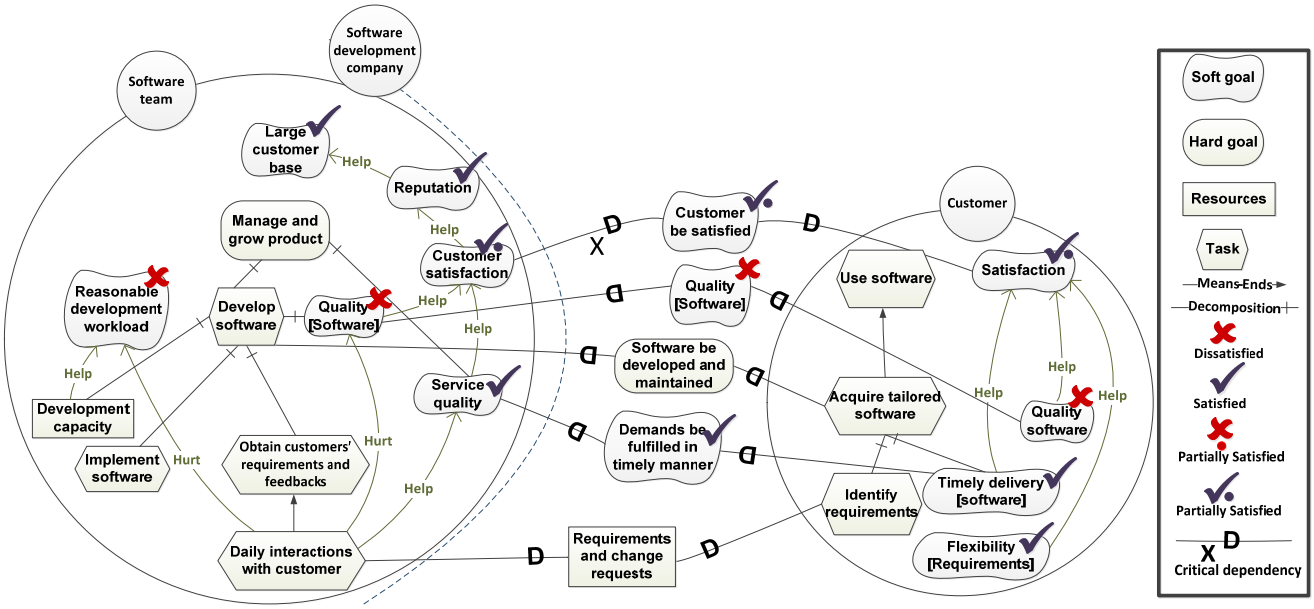

Fig. 1. Organziational Configuration of the Creative Chaos Stage

To this objective, we analyze the ethnography of a software firm [7] which has gone through four evolutionary stages of Creative Chaos, Waterfall, Agile Product Line Engineering, and Software Ecosystems over fourteen years, using *i**. At each stage, we focus on revealing the driving forces of change prompting evolution from one stage to another and how they trigger transition to a new organizational configuration. We utilize *i** models to uncover problems that trigger evolution and to support analysis on how to transition to the next stage.

At the border of each transition, four meta-steps are followed:

1) The snapshot of the actors' network before each transition is modeled based on the textual description of the scenario reported in [7].

2) The driving forces of change prompting evolution are captured in terms of internal and external unmet goals and quality criteria of the participant actors.

3) The *i** models are used to analyze and characterize the problematic situations and to reason about how to resolve it.

4) The snapshot of actors' configuration after each transition is modeled based on the textual description of the scenario.

A. The Creative Chaos Stage

Scenario Description: In its start-up phase, the main objective of the company was to develop and grow its homemade software product. To this end, the company started with a small software team that offered services to a small number of customers. The main driver of product development was daily interactions between customers and the developers. No specific restriction and control was applied on requirements and change requests. This phase was extremely productive and creative. However, with the growth of product and increase of customers, the number of features and functionalities grew as well as the number of defects. The development work became stressful with lots of overtime and little control [7].

Modeling and Analysis with *i:** To better understand and eventually resolve the problematic situation, one would need to know: What do specifically customers depend on the developers for? Why can't developers keep up with the development workload? Why does the development team interact daily with the customers? What other options does software team have to communicate with customers?

In the Creative Chaos stage (Fig. 1), customers depended on the software team to develop quality software in timely manner. This is modeled with the goal dependency of "Software be developed and maintained", and the soft goal dependencies of "Quality [software]" and "Demands be fulfilled in timely manner" between the "Customers" and the "Software Team" actors. On the other hand, for the software team and the organization, in its start-up stage, the satisfaction of the customers was of critical strategic importance (the X notation in Fig. 1). One reason for the importance of this dependency is that from business perspective, "Customer satisfaction" contributes to "Reputation" of the organization and leads to the increase of customers (the "Large customer base" soft goal). Therefore, "Customer be satisfied" is modeled as a critical soft goal dependency between software team and customer.

From a technical perspective, to "Develop software", the software team had to "Obtain customers' requirements and feedbacks". To this end, the software team interacted daily with the customers ("Daily interactions with customer" task). In these interactions, customers provided the software team with "Requirements and change Requests" – a strategic

resource dependency between software team and customers. Requirements and change requests were directly received and responded to by the developers in a timely manner. Therefore, the "Daily interactions with customers" task led to improving the "Service quality" soft goal which subsequently contributed positively to the "Customer satisfaction" soft goal (a business objective) and satisficed the soft goal dependency of "Demands be fulfilled in timely manner".

However, with the growth of software product as well as customer base (change in the business environment), "Daily interactions with customers" task became problematic: Directing customers' requirements to the developers without any control, and their immediate application by developers caused damage to the "Quality [software]" soft goal (modeled by hurt contribution). This task also created huge workload for developers, hurting the "Reasonable development workload" soft goal for the software team. Moreover, to "Develop software" while preserving "Reasonable Development workload", the software team required the needed capacity from among its developers (illustrated by "Development capacity" resource decomposition relationship). However, with the increase of customers, the "Development capacity" resource did not respond to the increasing demands and requirements of customers.

As the organizational configuration of Fig. 1 depicts, in the Creative Chaos stage, the company heavily relies on the software team to meet both its business and technical objectives. There are direct relationships between customers and the software team, and the "Daily interactions with customers" activity is a source of problem for two desired qualities "Reasonable development workload" and "Quality [software]". Therefore, to resolve the unmet soft goals of "Reasonable development workload" and "Quality software", the software team has to *1)* look for other alternatives for obtaining customers' requirements and feedbacks, *2)* the intense dependencies between "Software Team" and "Customers" have to be reconfigured, and *3)* development capacity has to be increased.

Indeed, in the next stage, the software development company reconfigured the dependencies between the software team and customers (change in the organizational configuration), reduced the responsibilities of software team and came up with other alternatives for "Obtaining customers' requirements and feedback".

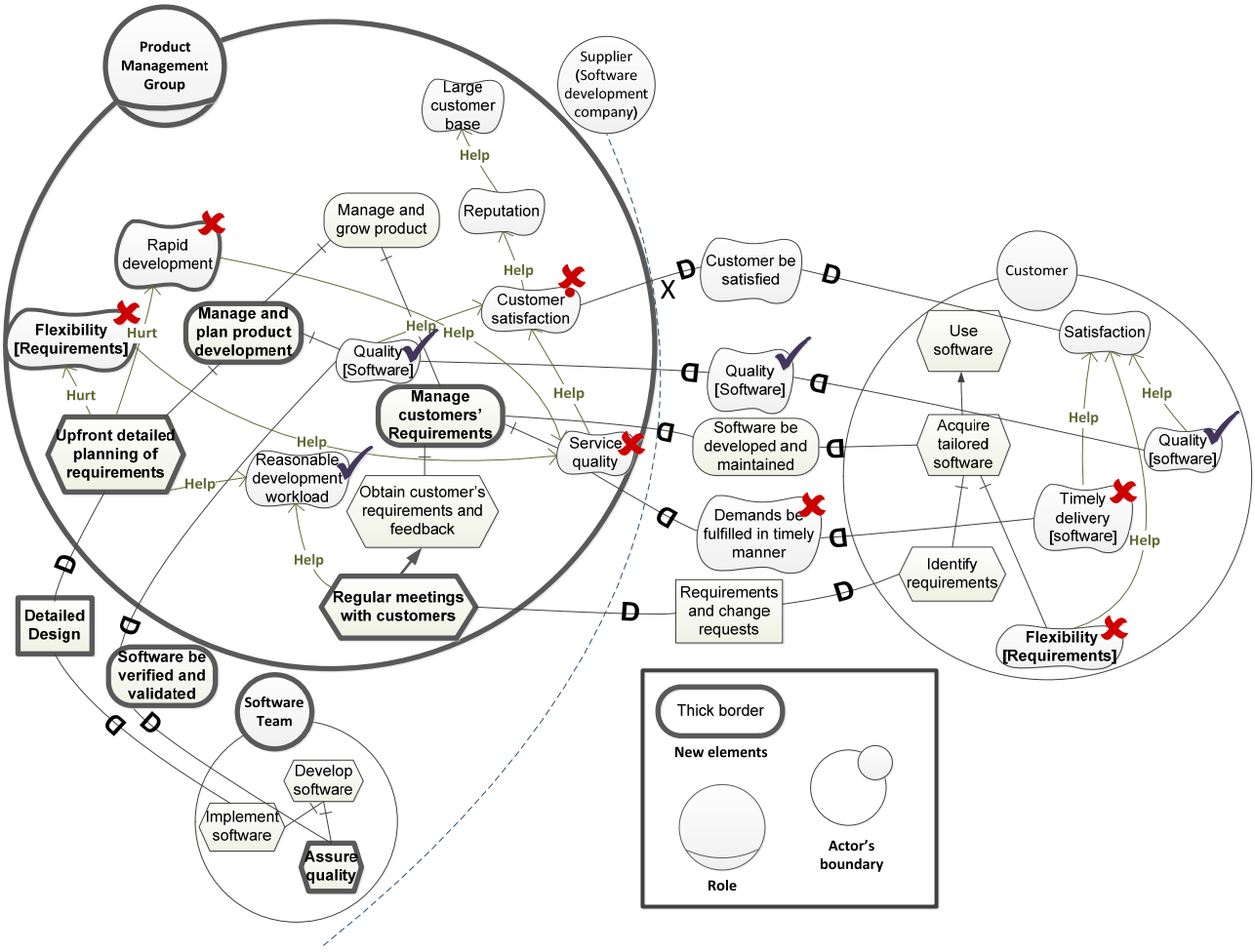

Fig. 2. Organziational Configuration of the Waterfall Stage

B. The Waterfall Stage

Scenario Description: To cope with the increase of customers, to manage the growth of product and of defects, and to control requirements and change requests, the company formalized the software development process according to waterfall development principles. Upfront detailed planning of requirements was thus emphasized followed by consecutive design and development processes. The Research and Development department (R & D) – in charge of developing software – was extended with quality assurance activities to establish a certain level of control, and to continue the development of the product alongside the growth of customer base. To manage the growth of product and the growth of customers, a Product Management Group was formed, comprising of a group of experienced employees. They have other responsibilities but spend part of their time in planning [7].

Modeling and Analysis with *i:** In terms of modeling, in this stage, we want to capture the reconfiguration of the organization and to see how it affected the objectives of the company and the customers including "Reasonable development workload", "Quality [software]" and "Demands be fulfilled in timely manner". Is it possible to show that "Development workload" has moderated in this stage?

As depicted in Fig. 2, to resolve the problem of unreasonable development workload, three main changes were applied in the organizational configuration of the development team: *1)* the Product Management Group (PMG) was formed, creating a new role with two main responsibilities: "Manage and plan product development" and "Manage customers and requirements". (To highlight the evolution of organizational configuration in comparison to the previous stage, the new elements are shown in bold font in the models.) PMG is modeled as a role since it can potentially be assigned to different persons. A group of experienced employees with other responsibilities partly played this role. Therefore, "Obtain customers' requirements and feedback" task was delegated to PMT as a sub task of "Manage customers and requirements". *2)* To obtain requirements from customers, "Daily interactions with customer" was replaced with "Regular meetings with customers" which contributed positively to the "Reasonable workload" soft goal of the PMG role. Moreover, to control requirements and regulate change requests, the task "Upfront detailed planning of requirements" was added to the responsibilities of the PMG role. The outcome of this task (the "Detailed design" resource) was fed as a resource into the "Implement software" task of "Software team". As Fig. 2 demonstrates, these changes in the configuration of work relationships significantly reduced the responsibilities of developers by delegating them to the PMG and satisfied the soft goal of "Reasonable development workload". *3)* Furthermore, to control the "Quality [software]" soft goal and to reduce defects, the "Assure quality" activity was added to the responsibilities of the software team.

However, this reconfiguration created severe problems for the company: "Upfront detailed planning of requirements" violated the "Flexibility" and "Rapid Development" soft goals, (related to the technical aspects of software development activities), which subsequently damaged the quality of service ("Service quality" soft goal was perverted). Decline of Service quality subsequently damaged the two soft goals of "Customer satisfaction" and "Demands be fulfilled in timely manners". Finally, this problem negatively influenced the "Customers' satisfaction" (the business aspect of the software development company). (Propagation rules in *i** [29] are utilized to reason over the negative influence of technical aspects on business objectives.) Moreover, postponing quality assurance activities to the end of the development process together with the slow production process resulted in the late identification of problems and increased rework. During the Waterfall stage, serious problems arose. Inflexibility of the development process on one hand, and instability of customers' requirements on the other, critically reduced production performance, incurring high costs on the organization.

C. The Agile Product Line Engineering Stage

Scenario Description: As the company grew, the product expanded into a product line capable of serving various usage scenarios. This change introduced two separate processes of development for the software products with different time cycles. Development of the product line was conducted in one-year cycles, while the development of customized products was conducted in much shorter cycles. Alongside the growth of the organization and of the product line, the Product Management Group was re-established as a full-time Product Strategy Group team managed by a Chief Strategy officer. The company acquired one of its former competitors and boosted the number of employees significantly. Through extensive internal training during one year, most of the organization was using the Agile development process. To improve the performance of the development and to cope effectively with changing users' requirements, Agile software development techniques were adopted. Functional requirements were replaced by explicit expression and evaluation of product qualities, which were preferably stated by customers involved in the development process. Moreover, regular customer review meetings were held in which the result of iterative development was evaluated and validated by customers. As a result, the number of issues near release was reduced, and the delivered product matched customers' expectations better [7].

Modeling and Analysis with *i:** In terms of modeling, one would need to ask: How did the software development organization attain flexibility in software development and software products? What choices and options did it create for its customers?

As depicted in Fig. 3, "To develop software", the "Iterative development" means was adopted by the software team which was decomposed into two main activities of "Develop product line" and "Develop customized products". This change introduced two new options: *1)* "Offer pre-configured products" (which offered ready-to-use products based on the common needs of customers) and *2)* "Offer customized products" (which offered products requiring customized configuration and minor development). As a result, the needs of the new customers were fulfilled immediately by offering

pre-configure products. Moreover, the specific needs and requirements of the special customers were addressed by configuring products in short cycles These two options alleviated the problems with the two soft goals of "Rapid development" and "Flexibility [Requirements]" to some extent.

To manage the development of the software product line, the PMT role turned into a full-time responsibility of the "Product Strategy Group" (depicted by actor element in *i**) with the main responsibility of "Strategic planning for software product line". The result of this activity was the "Product road map" to identify the main features to be added to the core of the product line, which was fed into the "Develop product line" process of the software development team.

To improve the performance of the development and to cope more effectively with changing requirements of users, Agile software development techniques were adopted. Notably, "Regular customer review meetings" were held in which the result of the iterative development were evaluated and validated by customers as a means to "Obtain Customer's requirements and feedback". As a result, the number of issues near release was reduced, and the delivered products matched customers' expectations better (the soft goals of "Service quality" and "Customer satisfaction" were satisficed.

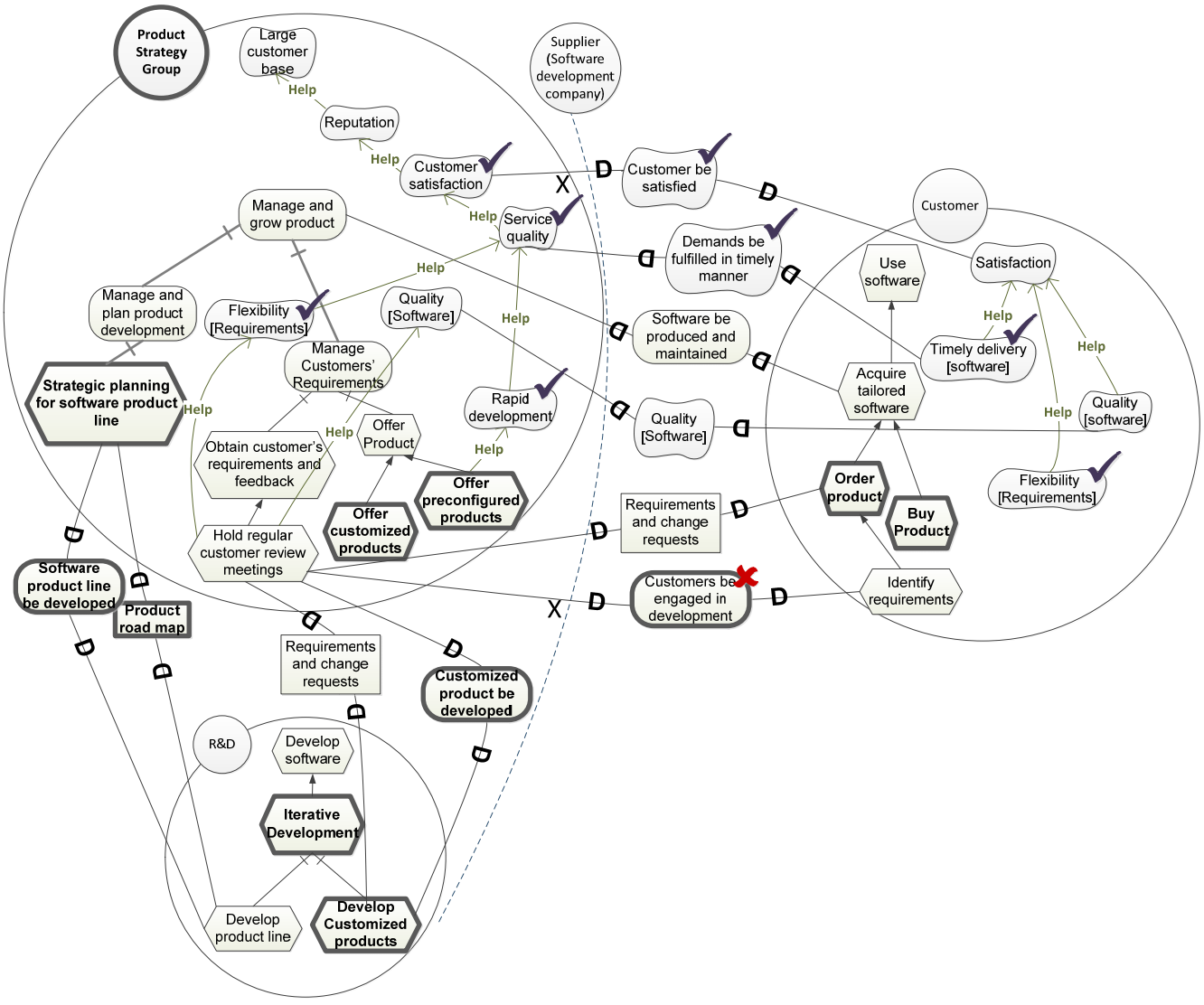

Fig. 3. Organziational Configuration of the Agile Product Line Engineering Stage

The adoption of the Agile development required active participation and engagement of customers in the development process. Therefore, it created a new critical goal dependency between the organization and the customers: "Customers be engaged in development" – marked by X in Fig. 3. However, this dependency required customers to regularly assign part of their time to product development. This expectation was not welcome by the customers because it required customers or customer representatives to spend some of their busy time in the development process of the company. In other words, they did not have enough motivation to participate. As depicted in Fig. 3, the "Customers be engaged in development" dependency was not met in this stage. Consequently, although the introduction of the Agile development resolved the problems of slow and flexible development, it created misalignment problems between the interests of the customers and objectives of the company.

From an analytical viewpoint, this raises the question of how this misalignment between the intents of the organization and the motivation of customers can be resolved. How can the customers be motivated and encouraged to participate in the development process in addition to fulfilling their own objectives? Is it possible to create a sense of being a member of the organization within the customers?

D. The Software Ecosystem Stage

Scenario Description: To foster customer participation, the company provided training programs to both developers and customers. After a few iterations, they became engaged in the development process. Co-creating the product became the common objective of the company and the stakeholders. On one side, the company had the most up-to-date knowledge of the technology and the ability to make use of it. On the other side, customers held the most up-to-date knowledge of various business domains and the domain-specific requirements of the product. To enhance supporting services for the software product, and to facilitate the integration of the software with other products, the company began offering extensibility frameworks and APIs. The provision of APIs provided external stakeholders with the business opportunity of extending the product with additional features and functionalities independently of the company. As a result, a third party was shaped; i.e., a set of external organizations based their business on the software product as a platform with the main objectives of developing value-adding solutions and products, and offering consulting services. To facilitate the extension of the core product by the third party, the company developed a flexibility framework as an interface for extending the software product line. This collaboration became beneficial for both sides. On one hand, the company could focus on the development and evolution of the core of software product line while delegating the development and extension of variants and plug-ins to the third party. On the other hand, the third party could have access to the big market of the company [7].

Modeling and Analysis with *i:** The model produced for this stage should explain how the misalignment between company's objectives and customers' interests was resolved in the Software Ecosystem stage. How were incentives created in the customers to entice them to participate in the development of the product? What changes happened in the relationships between customers and the organization?

As illustrated in Fig. 4, "Offer extensibility framework" was added to the set of services the company provided. As a result, two actors were introduced to the configuration of the organization. The Extensibility framework was added as a new technical actor to the socio-technical environment of the organization serving as an intermediary between the inner boundary of the organization and the external environment (see the placement of this actor on the border of Software development company actor in Fig. 4). Offering the extensibility framework created the "Third Party" as a new external stakeholder of the organization. The "Extensibility framework" actor provided the "Third Party" with the "Platform extension points" resource enabling them to "Extend software product services" in various ways – including "Develop value-adding products" and "Offer consulting services".

Provision of the extensibility framework contributed to the interests of the organization in two ways: *1)* It created the opportunity of "Open innovation" which had a positive influence on the "Reputation" soft goal. *2)* Via the extensibility framework, customers could integrate the software with other products. This change added "Variety of services" which had a positive contribution to the "Customer attraction" soft goal and contributed to "Large customer base". Moreover, this "Third Party" actor had the same interest in the product as the organization: Third party's business depended on the software product. Therefore, engagement in the development was beneficial to it. As a result, the dependency of "Customers be engaged in development" between the organization and customers was shifted to the third party and was satisficed since it was of "Shared value". In return, the third party was dependent on the company for it to become visible to a large base of the organization's customers ("Visibility to customers" soft goal), and to be provided with a "Marketing channel". Correspondingly, the responsibility of "Support third party" was added to the activities of the organization.

As Fig. 4 clearly demonstrates, the external environment of the company has dramatically changed in the Software Ecosystems stage. The addition of the Extensibility framework actor and its use as a software platform, has relatively opened the boundary of the organization to the third party and enabled the independent extension of the software product by actors external to the company. Moreover, the supply network of the software product has been distributed between the main organization and the third party.

IV. DISCUSSION

In this section, we first discuss our experience in attempting to analyze the evolution of a software development firm and highlight the challenging issues. Then, we examine to what extent the capabilities of the *i** modeling framework can help address some of these challenges.

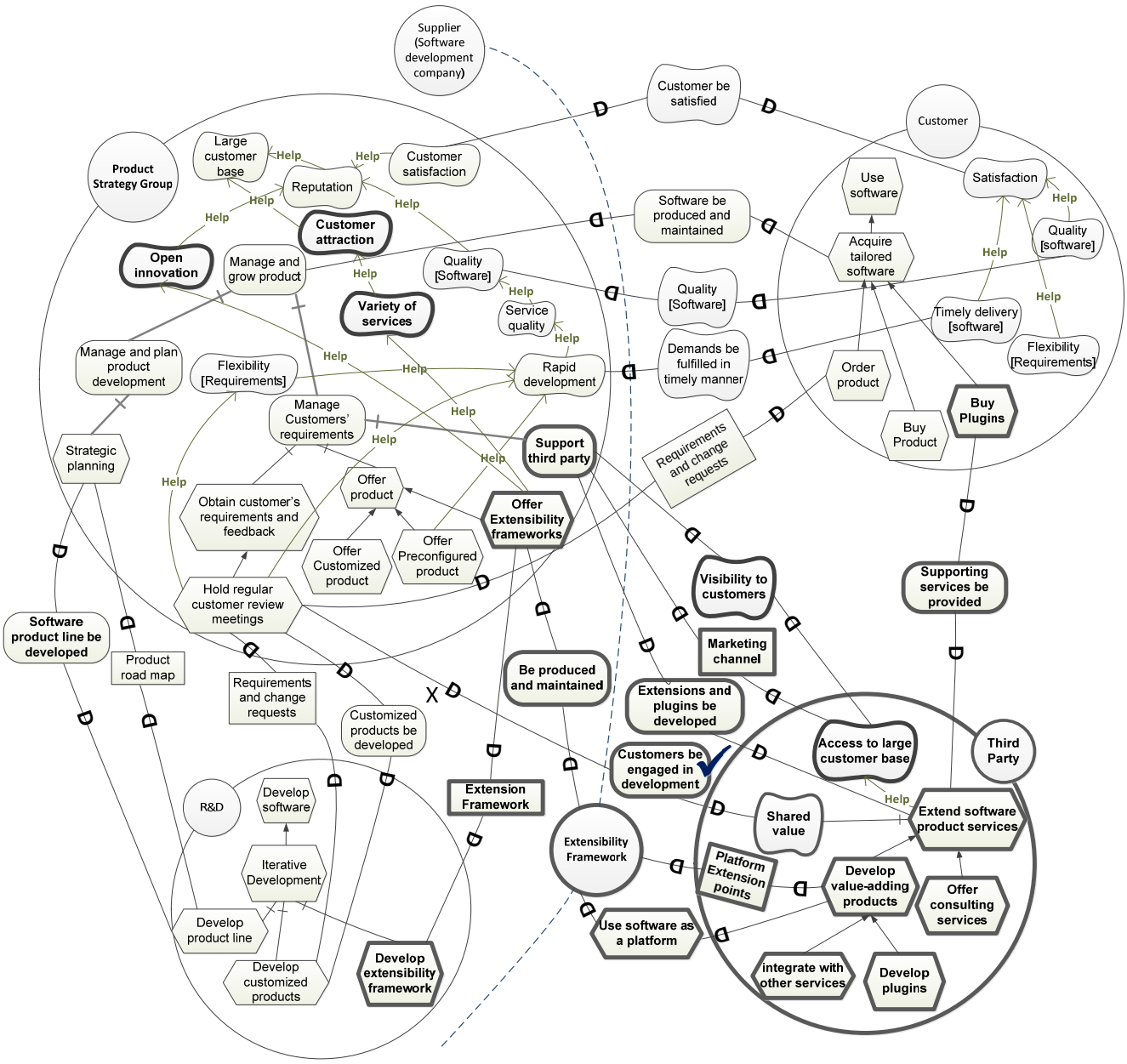

Fig. 4. Organziational Configuration of the Software Ecosystem Stage

A. *Observations*

Throughout analyzing the case study, we observed the following issues:

1) Evolution is a multi-dimensional phenomenon, not limited to the technical aspects of software development. In the software development organization under study, various business, organizational, social, and technical aspects co-evolved over relatively short periods. For example, in the start-up stage, changes in the business and organizational environment, such as the increase of customers, led to the reorganization of software development activities. In contrast, in the Software Ecosystems stage, evolution in software development technology (developing web APIs for the product line) created a new business environment which dramatically changed the stakeholders' relationships in the supply network of the software product, and confronted the organization with new business opportunities and threats.

Implications: The above examples provide evidence that: *a)* Business, social, and organizational environment change throughout software development over relatively short

periods; and *b)* Evolution of software development products and activities may dramatically change the business and organizational environment. Therefore, from the viewpoint of a software development firm, *i)* business, organizational and social aspects do not remain fixed throughout software development; and *ii)* evolution and adaptation are multi-dimensional, and not merely limited to software products and software processes.

Consequently, to have a clear understanding of the evolution of software development, it is necessary to consider the broader (business, organizational and social) environment in which software development is conducted. In other words, in addition to software-related causes, factors such as organizational, business and economic, and social aspects should also be considered as the causes of software evolution. Moreover, to manage this multi-dimensional evolution in a software development firm, systematic methods and analysis techniques are required which go beyond the level of software products and software processes in dealing with evolution and adaptation.

2) Change in software development activities is tightly tied with organizational design, business design and social design. In the case study, we came across several instances where social, business and organizational designs were required to be revised and reconfigured in tandem with change in software development activities. For example, one main driving force prompting transition from the Agile Product Line Engineering stage to the Software Ecosystems stage was the lack of enough motivation in the customers to collaborate in the development of software products – a social factor which was critical to the success of software development processes of the enterprise. However, this problem was resolved not by treating only the social aspects of software development activities. The problem was resolved by introducing a combination of changes in the technical and business aspects. Creating a new business environment, in which external stakeholders could have shared value (individual benefits) in software products, could attract the interest of third parties to participate. This change in the business environment was respectively enabled by the development of web APIs, which provided the technical infrastructure for new forms of business collaboration.

Implications: The above series of changes convey that: *a)* Evolution of software development activities is an ongoing enterprise development in which the analysis and design of social, organizational, and business factors is as crucial as software design and development and should be conducted in parallel; and *b)* The boundaries of software development are not limited to a single organization, a single project and a single software product. These issues can be observed *i)* in the evolution of software development throughout various stages of the organization; and *ii)* in the Software Ecosystems stage alone. In this stage, the boundaries of software development were opened up to external organizational actors, and the scope of software development has shifted to software platforms, transcending the temporal and spatial boundaries of a single software project.

Hence, systematic methods and analysis techniques are required to consider the issue of software development beyond the temporal and spatial scope of a single software project, a single software development firm and a single software product. Additionally, analysis and design of business and economics, social, and organizational configurations should be conducted in tandem with software development activities.

B. Support for Analysis and Modeling

In this section, we highlight some analysis and modeling requirements needed to address the challenges discussed in section A, and examine how *i** strategic actors modeling framework can address these requirements.

- *Reasoning about the co-evolution of software development with organizational, business, and social designs.* To address this type of analysis, it is required *a)* to capture and identify the causes of evolution in the socio-technical context of software development, and *b)* to identify and reason about the influence of these causes on each other and on the evolution of software development.
- *Co-aligning software design and development with business, organizational and social designs.* To address this design concern, it is required *a)* to capture and model various heterogeneous aspects, such as software development activities, organizational configurations, business environment, and social aspects in a single model; *b)* to elicit and reason about the dependencies and relationships between these heterogeneous aspects; and *c)* to reason about the alternatives and trade-offs to transition from one configuration to another.

In the following, we discuss how the *i** modeling framework addresses the above requirements:

1) Homogenous modeling for the heterogeneous context of a software development organization.

*i** offers two mechanisms to address this requirement: *a)* By modeling the socio-technical context of a software development organization as a network of strategic actors, it provides the uniform element of strategic actor for modeling different social, business, organizational and technical elements, and the dependencies between these elements. For example, in the Software Ecosystems stage, the notion of actors can be used to model both the human actors such as "Customer" and "Third Party" as well as "Extensibility Framework" which is a technical actor in the configuration of the organization. *b)* Goal decomposition relationships and contribution relationships between soft goals can be used to capture and analyze a collection of heterogeneous elements together. For example, in the Software Ecosystems stage (Fig. 4) soft goals such as "Reputation" and "Customer attraction" are business objectives. However, goals such as "Rapid development" and "Quality [software]" are objectives related to the software development activities. *i** models enable capturing and analysis of the collection of these factors in one model.

2) Strategic & intentional reasoning about evolution.

The notion of goals in *i** (hard goals and soft goals) can be used to provide an intentional perspective of evolution in software development. More precisely, the reasons for

evolution can be captured in terms of misalignment between the goals and quality objectives of strategic actors and the current configuration of the organization (unmet goals and unmet soft goals of the strategic actors in the as-is situation). For example, one driving force of change prompting transition from the Creative Chaos to the Waterfall stage was that the organizational configuration of the firm was not aligned with the "Reasonable development workload" soft goal of the "Software team" actor. This misalignment is captured in terms of an unmet quality objective in Fig. 1. Another misalignment prompting the evolution was the unmet soft goal "Quality [software]" (a critical strategic dependency between "Software team" actor and "Customer" actor) in the configuration of the Creative Chaos stage. Throughout the analysis of the case study, we developed an intentional reasoning over the evolution of the software development organization conforming to how the evolutionary trajectory of the firm unfolded over time.

3) Open-ended means-ends relationships to model and reason about alternative evolutionary intentional actions.

The notion of open-ended intentional variability in means-ends relationships can be utilized to model and reason about the alternative evolutionary strategic courses of actions that can be taken by strategic actors. If the driving forces of change are captured in terms of unmet quality objectives, strategic actors can come up with new means (including new tasks, resources, goals and strategic dependencies) to satisfy them. For instance, one change in the activities of the organization from the Waterfall to the Agile Product Line Engineering (Fig. 3) was that new alternatives were introduced to achieve "Offer product" task; i.e., in addition to "Offer customized product", a new option was created to "Offer pre-configured products", which could control one portion of the demands of new customers. Respectively, as the company evolved, new options and variabilities were introduced for offering products to customers. Moreover, in some cases, the adoption of a new means requires establishing new dependencies with existing or new actors. For example, in the Software Ecosystem stage, the addition of the "Offer extensibility framework" as a new means to address the "Manage customers' requirements" end resulted in establishing new relationships between the company and the third party. This kind of change results in the evolution of social and organizational configuration of the firm.

4) Reasoning about social, organizational and business designs together with technical design.

a) Designing the social system around software development. The notion of strategic actors can be used to analyze the social design of an organization. For example, in the Agile Product Line Engineering stage, one driving force of change was the unmet strategic dependency of "Customers be engaged in development". This unmet goal reveals the misalignment between the intents of software development organization and the interests of the customers. The analysis of how to assure that customers have "Shared value" (as a soft goal) in the development of software product is one attempt in redesigning the social system of the enterprise in order to motivate customers to collaborate with the organization.

b) Designing the organizational configurations around software development. The concepts of actor boundary, strategic dependencies, and roles can be utilized to analyze the design of organizational configurations, and distribution of responsibilities and activities between organizational actors in a software development enterprise. For example, In the Creative Chaos stage, one problem was the heavy workload of the software development team. Using *i** modeling, one could analyze and demonstrate how the organizational configuration of the firm was rearranged to resolve this problem. The reduced number of strategic dependencies between the "Software Team" actor and other organizational actors as well as the reduced number of tasks of the "Software Team" actor in Fig. 2, clearly demonstrate resolution of the problem along with the growth of the organizational structures.

c) Designing the business environment around software development. The concept of strategic rationale – including goals, means-ends relationships, and quality objectives – can be used to help analyze the governance of a software development organization (decisions about strategies, tactics and operations of a software development firm [30]) and its business dynamics. For instance, in the Software Ecosystem stage, the company had to decide how to open up the organizational boundaries toward external stakeholders. This decision involves multiple levels of analysis about: *i)* how to incorporate external organizations and users in the business of the software development firm (strategic level); *ii)* how to open up the product line to external stakeholders (tactical level); and *iii)* how to open up source code to external developers (operational level). To take adaptive decisions, the software development company should first have a clear understanding of the strategic objectives and dependencies of the involved stakeholders. Second, it needs to analyze the benefits and costs as well as the opportunities and threats associated with each alternative course of action it can take. To reason about how to design the relationships between different stakeholders, the *i** notion of strategic dependencies and actor boundaries can be used. Moreover, to explicate and analyze the alternatives and trade-offs related to each of the above levels of decision making, the *i** notion of strategic rationale can be used in addition. *i**'s support for adaptive organizational governance and business dynamics of a software development firm has been further discussed in [31].

V. CONCLUSION

Our experience analyzing scenarios of a software development organization shows that: *a)* Software development activities co-evolve with social, business, and organizational environment over short periods; *b)* Software development is tightly tied with business, social, and organizational designs; and *c)* The scope of software development transcends the boundaries of a single software development organization, a single software development project, and a single software product. These features are evident both from the evolution of the software development organization over several stages and the specific stage of Software Ecosystem, which is the most recent trend in the

software industry. Therefore, to help a software development firm (and subsequently a software product) develop and grow sustainably, analysis techniques and systematic engineering methods are required which address the multi-dimensional co-development of software products and software activities together with business and economics, organizational configurations, and social aspects. However, currently, the focus of software development approaches is mainly on the engineering and adaptation of a single software system within the scope of a single project and a single organization, neglecting the above issues [6, 32].

To address the above shortcomings, in a preliminary attempt, we modeled and analyzed the evolutionary trajectory of a software development firm, taking into account the organizational, business and social dimensions as well as the software product and software development activities. We also delineated how capabilities of the *i** strategic actors modeling framework can help analyze, design and align various heterogeneous dimensions of a software development firm. Nevertheless, supporting the sustainable development and evolution of a software development firm – specifically in the face of recent trend towards Software Ecosystems – is a much more complicated issue than what was discussed herein. We aim to explore this research agenda further in our ongoing research.

ACKNOWLEDGEMENTS

Financial support from the Natural Sciences and Engineering Research Council of Canada (NSERC) and the Ontario Research Foundation (ORF) are gratefully acknowledged.

REFERENCES

[1] B. Boehm, and R. Turner. Balancing agility and discipline: A guide for the perplexed. Addison-Wesley Professional, 2003.

[2] M. Iansiti, and R. Levien. "Strategy as ecology." Harvard business review82, no. 3 (2004): 68-81.

[3] D. Messerschmitt, and C. Szyperski. "Software ecosystem: understanding an indispensable technology and industry." MIT Press Books 1 (2005).

[4] J. Bosch. "From software product lines to software ecosystems." InProceedings of the 13th International Software Product Line Conference, pp. 111-119. Carnegie Mellon University, 2009.

[5] S. Jansen, A. Finkelstein, and Sjaak Brinkkemper. "A sense of community: A research agenda for software ecosystems." In Software Engineering-Companion Volume, 2009. ICSE-Companion 2009. 31st International Conference on, pp. 187-190. IEEE, 2009.

[6] M. Jarke, P. Loucopoulos, K. Lyytinen, J. Mylopoulos, and W. Robinson. "The brave new world of design requirements." Information Systems 36, no. 7 (2011), pp. 992-1008.

[7] G. K. Hanssen, "A longitudinal case study of an emerging software ecosystem: Implications for practice and theory." Journal of Systems and Software 85, no. 7 (2012), pp. 1455-1466.

[8] E. Yu, P. Giorgini, N. Maiden, and J. Mylopoulos, (Eds.). (2011). Social modeling for requirements engineering. Mit Press.

[9] N. Chapin, J. Hale, K. Khan, J. Ramil, and W. Tan. "Types of software evolution and software maintenance." Journal of software maintenance and evolution: Research and Practice 13, no. 1 (2001), pp. 3-30.

[10] M. Lehman, J. Ramil, P. Wernick, D. Perry, and W. Turski. "Metrics and laws of software evolution-the nineties view." In Software Metrics Symposium, 1997. Proceedings., Fourth International, pp. 20-32. IEEE, 1997.

[11] N. Madhavji, J. Fernandez-Ramil, and D. Perry, eds. Software evolution and feedback: Theory and practice. John Wiley & Sons, 2006.

[12] T. Men, A. Serebrenik, and A. Cleve, eds. Evolving Software Systems. Springer Berlin, 2014.

[13] M. Lehman, and J. Ramil. "Software evolution and software evolution processes." Annals of Software Engineering 14, no. 1-4 (2002), pp. 275-309.

[14] T. Mens, M. Wermelinger, S. Ducasse, S. Demeyer, R. Hirschfeld, and Mehdi Jazayeri. "Challenges in software evolution." In Principles of Software Evolution, Eighth International Workshop on, pp. 13-22. IEEE, 2005.

[15] S. Bandinelli, A. Fuggetta, and C. Ghezzi. "Software process model evolution in the SPADE environment." Software Engineering, IEEE Transactions on 19, no. 12 (1993), pp. 1128-1144.

[16] E. Yu, A. Lapouchnian, and S. Deng. "Adapting to Uncertain and Evolving Enterprise Requirements: The case of business-driven business intelligence." In Research Challenges in Information Science (RCIS), 2013 IEEE Seventh International Conference on, pp. 1-12. IEEE, 2013.

[17] B. Cheng, R. Lemos, H. Giese, P. Inverardi, J. Magee, J. Andersson, B. Becker et al. "Software engineering for self-adaptive systems: A research roadmap." In Software engineering for self-adaptive systems, pp. 1-26. Springer Berlin Heidelberg, 2009.

[18] P. Oreizy, N. Medvidovic, and R. N. Taylor. "Architecture-based runtime software evolution." In Proceedings of the 20th international conference on Software engineering, pp. 177-186. IEEE Computer Society, 1998.

[19] J. Highsmith. Adaptive software development: a collaborative approach to managing complex systems. Addison-Wesley, 2013.

[20] R. Bendraou, B. Combemale, Xavier Crégut, and M-P. Gervais. "Definition of an Executable SPEM 2.0." In Software Engineering Conference, 2007. APSEC 2007. 14th Asia-Pacific, pp. 390-397. IEEE, 2007.

[21] V. Boucharas, S. Jansen, and S. Brinkkemper. "Formalizing software ecosystem modeling." In Proceedings of the 1st international workshop on Open component ecosystems, pp. 41-50. ACM, 2009.

[22] T. Gîrba, and S. Ducasse. "Modeling history to analyze software evolution." Journal of Software Maintenance and Evolution: Research and Practice 18, no. 3 (2006), pp. 207-236.

[23] C. Seidl, and U. Aßmann. "Towards modeling and analyzing variability in evolving software ecosystems." In Proceedings of the Seventh International Workshop on Variability Modelling of Software-intensive Systems, p. 3. ACM, 2013.

[24] V. Rajlich. "Modeling software evolution by evolving interoperation graphs."Annals of Software Engineering 9, no. 1-2 (2000), pp. 235-248.

[25] T. Mens, and M. Goeminne. "Analysing the evolution of social aspects of open source software ecosystems." In Third International Workshop on Software Ecosystems (IWSECO-2011), pp. 1-14. 2011.

[26] T. Mens, M. Claes, P. Grosjean, and A. Serebrenik. "Studying Evolving Software Ecosystems based on Ecological Models." InEvolving Software Systems, pp. 297-326. Springer Berlin Heidelberg, 2014.

[27] R. Madachy. Software process dynamics. John Wiley & Sons, 2007.

[28] B. Chatters, M. Lehman, J. Ramil, and P. Wernick. "Modelling a software evolution process." Software Process: Improvement and Practice 5, no. 2-3 (2000), pp. 91-102.

[29] J. Horkoff, and E. Yu. "A qualitative, interactive evaluation procedure for goal-and agent-oriented models." In CAiSE Forum. 2009.

[30] S. Jansen , S. Brinkkemper, J. Souer, and L. Luinenburg. "Shades of gray: Opening up a software producing organization with the open software enterprise model." Journal of Systems and Software 85, no. 7 (2012), pp. 1495-1510.

[31] E. Yu, and S. Deng. "Understanding Software Ecosystems: A Strategic Modeling Approach." In Proceedings of the Workshop on Software Ecosystems, pp. 65-76. 2010.

[32] Y. Dittrich. "Software Engineering Beyond the Project–Sustaining Software Ecosystems." Information and Software Technology (2014), doi: 101016/j.infsof.2014.02.012.