Am I going to Heaven? First step climbing the Stairway to Heaven Model

Results from a Case Study in Industry

Paulo Sérgio dos Santos Júnior
Department of Computer Science, Federal Institute of Education, Science and Technology of Espírito Santo
Serra, ES, Brazil
paulo.junior@ifes.edu.br

Monalessa P. Barcellos
Ontology and Conceptual Modeling Research Group (NEMO), Computer Science Department, Federal University of Espírito Santo
Vitória, ES, Brazil
monalessa@inf.ufes.br

Rodrigo Fernandes Calhau
Department of Computer Science, Federal Institute of Education, Science and Technology of Espírito Santo
Serra, ES, Brazil
calhau@ifes.edu.br

ABSTRACT

Context: Nowadays, software development organizations have adopted agile practices and data-driven software development aiming at competitive advantage. Moving from traditional to agile and data-driven software development requires changes in the organization culture and structure, which may not be easy. Stairway to Heaven Model (StH) describes this evolution path in five stages. *Objective*: We aimed to investigate how Systems Theory tools, GUT Matrix and reference ontologies can help organizations in the first transition of StH, i.e., moving from traditional to agile development. *Method*: We performed a participative case study in a Brazilian organization that develops software in partnership with a European organization. We applied Systems Theory tools (systemic maps and archetypes) to understand the organization and identify undesirable behaviors and their causes, and also GUT matrices to decide about which ones should be addressed first; we defined strategies to change the undesirable behaviors by implementing agile practices, and we used reference ontologies to share a common understanding about agile concepts. *Results*: By understanding the organization, a decision was made to implement a combination of agile and traditional practices. The implemented strategies improved software quality and project time and cost. Problems due to misunderstanding agile concepts were solved, allowing the organization to experience agile culture and foresee changes in its business model. *Conclusion*: Systems Theory tools and GUT Matrix aid organizations to move from traditional to agile development by supporting better understand the organization, find leverage points of change and enabling to define strategies aligned to the organization characteristics and priorities. Reference ontologies can be useful to establish a common understanding about agile, enabling teams to be aware of and, thus, more committed to agile practices and concepts.

KEYWORDS

Stairway to Heaven, Agile, Systems Theory, GUT Matrix, Ontology

SBES 2020, October 19–23, 2020, Natal,RN - Brazil

ACM ISBN 978-1-4503-8753-8/20/09…$15.00
https://doi.org/10.1145/3422392.3422406

ACM Reference Format:
Paulo Sérgio dos Santos Júnior, Monalessa P. Barcellos, and Rodrigo Fernandes Calhau. 2020. Am I going to Heaven? First step climbing the Stairway to Heaven Model: Results from a Case Study in Industry. In *34th Brazilian Symposium on Software Engineering (SBES '20), October 21–23, 2020, Natal, Brazil.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3422392.3422406

2020 SBES '20,October 21–23, 2020, Natal, Brazil 978-1-4503-8753-8/20/09

1 INTRODUCTION

Typically, fast-changing and unpredictable market needs, complex and changing customer requirements, and pressures of shorter time-to-market are challenges faced by organizations. To address these challenges, many organizations have started adopting agile development methods with the intention to enhance the organization ability to respond to change. In emphasizing flexibility, efficiency and speed, agile practices have led to a paradigm shift on how software is developed [12] [21]. Different flavors of the agile methods have become the *de facto* way of working in the software industry [14]. In allowing for more flexible ways of working with an emphasis on customer collaboration and speed of development, agile methods help organizations address many of the problems associated with traditional development [5].

The adoption of agile practices has enabled organizations to shorten development cycles and increase customer collaboration. However, this has not been enough. There has been a need to learn from customers also after deployment of the software product. This requires practices that extend agile practices, such as continuous deployment (i.e., the ability to deliver software more frequently to customers and benefit from frequent customer feedback), which enables shorter feedback loops, more frequent customer feedback, and the ability to more accurately validate whether the developed functionalities correspond to customer needs and behaviors [13]. Therefore, organizations should evolve from traditional development towards data-driven software development.

Considering that organizations struggle with the changes to be made along the path and with the order in which to implement them, Olsson et al. [12] proposed the Stairway to Heaven Model (StH), which describes the typical successful evolution of an organization from traditional to customer data-driven development. The model comprises five stages, where the first transition consists in moving from traditional to agile development. This transition requires a

careful introduction of agile practices, a shift to small development teams, and a focus on features rather than components.

In this paper, we report the experience of a Brazilian organization (here called Organization A for anonymity reasons) which decided to evolve from traditional to agile and data-driven software development. For that, we have followed the StH model [12]. In this paper our focus is on the first transition of the StH model. Although there is an increasing number of organizations moving from traditional to agile, implementing the changes needed to the first transition prescribed in StH is not trivial because it involves changes not only in the development process, but also in the organization culture. Moreover, there is no "one and right" way to implement agile practices in an organization because each agile practice has to be tailored to fit the business goals, culture, environment and other aspects of the organization. Therefore, organizations should find their own way to go through the path from traditional to agile [8].

Organization A has a particular characteristic that needs to be taken into account when defining strategies to implement agile practices: the software projects of Organization A are built in partnership with a European organization (here called Organization B). In this partnership, Organization B is responsible for software requirements specification process, while Organization A is responsible for design, coding, testing and deployment processes. Furthermore, Organization B is responsible for the communication between Organization A and the project client. Both organizations A and B work in a traditional but many times ad hoc manner. This way of working has brought problems, such as budget overloading, teams divided into disciplines (testers, architects, programmers, etc.) causing many intermediary delivery points in the organization and increasing delays between them, and large periods required to deploy new versions of the software products [8] [13] [21].

Organization A was in the first stage of StH and, in order to evolve, the first step was to go towards becoming an agile organization. Two main challenges were faced in this context: (i) how to move from a traditional development culture to an agile culture and (ii) how to implement agile practices in an organization that shares requirement-related activities with another organization and does not have direct access to the project client.

To overcome these challenges, it would be necessary to get to know the organization so that it would be possible to define suitable strategies to implement agile practices. Thus, we employed an approach that combined Systems Theory tools (mainly systemic maps and archetypes) [11][17], GUT Matrix [9] and reference ontologies [7] to identify the path to implement agile practices and get into agile culture based on the organization characteristics and context. As main results we highlight: (i) it was possible to understand the organization behavior, identify behavior patterns and leverage points of change; (ii) strategies were defined to implement agile practices by changing undesirable behaviors and focusing on leverage points, taking the organization characteristics into account; (iii) by implementing the strategies, Organization A improved software quality, project time and cost and started to develop agile culture; and (iv) a process based on Systems Theory to aid organization define strategies to implement agile practices arose from the study.

This work brings contributions to researchers and practitioners. The study can serve as an example for other organizations similar to Organization A and the process resulting from the study can be used by other organizations. Researchers can reflect and provide advances on the use of Systems Theory to support definition of strategies in agile software development context.

This paper is organized as follows: Section 2 presents the theoretical background; Section 3 presents the study planning, execution and results; Section 4 discusses threats to validity and Section 5 presents our final considerations and future works.

2 BACKGROUND

Traditional software development is organized sequentially, handing over intermediate artifacts (e.g., requirements, designs, code) between different functional groups in the organization. This cause many handover points that lead to problems such as time delays between handovers of different groups and amounts of resources are applied to creating these intermediate artifacts that, to a large extent, are replacements of human-to-human communication [2]. In agile software development, the notion of cross-functional, multidisciplinary teams plays a central role. These teams have the different roles necessary to take a customer need all the way to a delivered solution. Moreover, the notion of small, empowered teams, the backlog, and daily stand up meetings and sprints guide software development through shorter cycles and help bring the software development closer to the client [2].

Moving from traditional to agile development is the first transition prescribed in Stairway to Heaven Model (StH) [12]. StH describes the evolution path organizations follow to successfully move from traditional to data-driven software development. It comprises five stages: traditional development, agile organization, continuous integration, continuous deployment, and R&D as an innovation system. In a nutshell, organizations evolving from traditional development start by experimenting with one or a few agile teams. Once these teams are successful, agile practices are adopted by the organization. As the organization starts showing the benefits of working agile, system integration and verification becomes involved and the organization adopts continuous integration. Once it runs internally, lead customers often express an interest to receive software functionality earlier than through the normal release cycle. They want continuous deployment of software. The final stage is where the organization collects data from its customers and uses a customer base to run frequent feature experiments to support customer data-driven software development [13].

Many organizations have moved from traditional to agile. There are many ways of doing that and each organization should consider its business goals, culture, environment and other aspects to find the best way to go through the path. In the experience reported in this paper we have used Systems Theory tools, GUT Matrix and reference ontologies, which are briefly introduced in the following.

A. Systems Theory: It has been used in industry and Academy to support (re)design of organizations [11][17][18]. It sees organization as a system, consisting of elements (e.g., teams, artifacts, policies) and interconnections (e.g., the relation between the development team, the software artifacts it produces and the policies that influence their production) coherently organized in a structure that produces a characteristic set of behaviors, often classified as its function or

purpose (e.g., the development team produces a software product aiming to accomplish its function in the organization)[11].

In the Systems Theory literature there are several tools that support understanding the different elements and behaviors of a system, such as systemic maps and archetypes[11][17]. A systemic map (also known as causal loop diagram) allows representing the dynamics of a system by means of the system borders, relevant variables, their causal relationships and feedback loops. A positive causal relationship means that two variables change in the same direction (e.g., increase the number of bad design decisions causes increasing in software defects), while a negative causal relationship means that two variables change in opposite directions (e.g., increase test efficacy causes decreasing in software defects). Feedback loops are mechanisms that change variables of the system. There are two main types: balancing and reinforcing feedback loops. The former is an equilibrant structure in the system and is source of stability and resistance to change. The latter compounds change in one direction with even more change.

One beneficial effect of using systemic maps is that they help identify archetypes. An archetype is a common structure of the system that produces a characteristic pattern of behavior. For example, the archetype Shifting the Burden occurs when a problem symptom is "solved" by applying a symptomatic solution, which diverts attention away from a more fundamental solution [10]. Each archetype has a corresponding modeling pattern. Therefore, by analyzing a systemic map is possible to identify archetypes by looking for their modeling patterns. Archetypes and systemic maps can be useful to identify problems and possible leverage points to solve them. Leverage points are points in the system where a small change can lead to a large shift in behavior[11].

B. GUT Matrix: It allows to prioritize the resolution of problems, considering that resources are limited to solve them [9]. The prioritization is based on: Gravity(G), which describes the impact of the problem on the organization; Urgency(U); referring to how much time is available to address the problem; and Tendency(T), which measures the predisposition of a problem getting worse over time.

C. Reference Ontology: Ontologies have been recognized as important instruments to solve knowledge-related problems. An ontology is a formal, explicit specification of a shared conceptualization [19]. Ontologies can be developed for communication purposes (reference ontologies) or for computational solutions (operational ontologies). A reference ontology is a special kind of conceptual model representing a model of consensus within a community. It is a solution-independent specification with the aim of making a clear and precise description of the domain in reality for the purposes of communication, learning and problem-solving. [1].

Some works have reported the use of Systems Theory in the agile development context. For example, Vighen and Wang [20] proposed a framework based on the Systems Theory that identifies enablers and inhibitors of agility, and discuss capabilities that should be present in an agile team. Gregory et al. [6] discuss challenges to implement agile and suggest some organizational elements that could be used to do that. Considering the StH context, Karvonen et al. [8] used BAPO categories to identify some practices to each StH step. However, they do not discuss how to understand the organization to establish proper strategies to implement them. Considering scenarios involving more than one organization to produce software, Sousa et al.[3] discuss agile transformation in Brazilian public institutions. Different from Organizations A and B, which work together to produce software to the client, Brazilian public institutions hire software organizations to develop software (i.e., the public institution is client of the hired organization). Moreover, different from the scenario discussed in[3], in our study, Organization A needed to develop skills, processes and culture that enabled it to work with multicultural issues, because Organization A, Organization B and clients are in different countries and have different cultures. None of the aforementioned works use Systems Theory tools, GUT matrix and reference ontologies to help organizations to define strategies to agile practices, as we did in our study.

3 CASE STUDY PLANNING, EXECUTION AND RESULTS

Participative case study was selected as research method in this study because two researchers acted as consultants in Organization A and, this way, were participants in the process being observed [1]. Together with other participants, they gathered information to understand the organization and defined strategies to implement agile practices. Thus, the researchers had some control over some intervening variables.

3.1 Study Design

3.1.1 Diagnosis. Organization A is a Brazilian software development organization that works together with a European organization (Organization B) to develop software products to European clients. It has 30 developers organized in teams managed by tech leaders. Organization B elicits requirements with clients and Organization A is in charge of developing the corresponding software. As a consequence of the increasing number of projects and team members, added to the lack of flexible processes, some problems emerged, such as projects late and over budget, increasing in software defects, overloading of the teams due to rework on software artifacts, and communication issues among client, Organization A and Organization B.

Aiming to minimize these problems, in the first semester of 2019, Organization A decided to implement Scrum practices, but without success. According to the directors, the main difficulties were due to non-direct communication with the client and included: difficult to define product backlog, select a product owner and carry out Scrum ceremonies that need client's feedback. Furthermore, they pointed out that agile culture demands knowledge and its clients, business partners and developers were not prepared for it. This scenario indicated to us that the particular characteristics and context of the organization had not been considered in the first try to implement agile practices. Therefore, at the beginning of 2020, we proposed to use StH as a reference model to evolve Organization A from traditional to data-driven software development, in a long-term process improvement program. The first step: move from traditional to agile. Considering the peculiar scenario of Organization A, we decided to use Systems Theory to understand the organization in a systemic way. Then, we used GUT Matrix to support prioritization

of problems resolution, and reference ontologies to provide common knowledge about agile development.

3.1.2 Planning. The study goal was to analyze the use of Systems Theory tools (particularly systemic maps and archetypes), GUT Matrix and reference ontologies to help define strategies to implement agile practices when the organization is moving from traditional to agile development. By strategies we mean actions or plans established to implement agile development. Aligned to this goal, the following research question was defined: *are Systems Theory, GUT Matrix and reference ontologies useful to define suitable strategies for an organization to move from traditional to agile development?*

The expected outcomes were: (i) a view of important aspects of the organization by means of systemic maps; (ii) prioritization of problems and causes to be addressed; (iii) strategies to address problems and implement agile practices; (iv) a Systems Theory-based process to define strategies to move from traditional to agile.

The study participants who directly participated in interviews to data collection and results evaluation were the two directors (software development director and sales director), one tech lead and two developers. The first and third authors worked as consultants in Organization A and, thus, also participated in the study.

3.2 Study Execution and Data Collection

3.2.1 Data Collection. Data collection involved interviews, development of systemic maps and GUT matrix, and definition of strategies to implement agile practices.

A. Initial Interviews. Data collection started with interviews to gather general information about the organization. Six interviews were conducted, four with the directors and two with the developers and tech leader. Participants were told to feel free to talk as much as they wanted to. Each interview lasted about 90 minutes. The funnel questions technique was used, i.e., the interview started with general questions (e.g., "What kind of software does the organization develop?", "How is the software development process?"), and then went deeper into more specific points of each one (e.g., "Tell me more about the software test activity"). The interviews were recorded, transcribed and validate with each participant.

The interviews with the directors aimed to get information about the following aspects: organizational environment, culture, rules of relationship with partners, future plans, software development process, software development issues and agile knowledge. Among the information provided by the directors, they pointed out that some problems were caused by misunderstood software requirements or project scope not clearly defined. According to them, Organization B did not describe requirements in a consistent and clear way.

The interviews with the tech leader and developers aimed at understanding software development problems under their perspective and how familiar they were with agile methods and practices. The problems mentioned by the directors were also reported by the tech leader and developers. When asked about team organization, they pointed out that the teams were not self-organized. Contrariwise, tech leaders were responsible for allocating tasks, coordinating team members, establishing deadlines and monitoring projects. Moreover, the team knowledge of agile was limited.

B. Systemic Maps. Information obtained in the interviews was used to build systemic maps. Figure 1 shows a fragment of one of the developed systemic maps. The elements in blue in the figure form a modeling pattern that reveals the presence of the archetype Shifting the Burden.

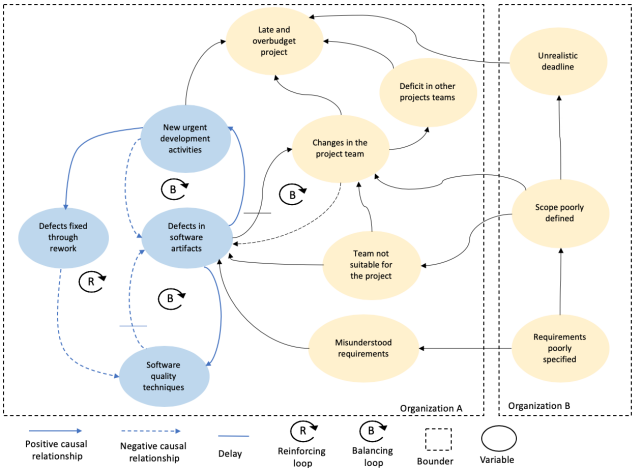

Figure 1: Fragment of systemic map

As previously said, Organization B is responsible for eliciting requirements with the client, specifying and sending them for Organization A to develop the software. The development teams of Organization A often misunderstand requirements that describe the software, component or functionality to be developed, since Organization B produces *Requirements poorly specified*, neither adopting a technique nor following a pattern to describe them.*Misunderstood requirements* contribute to increase the number of *Defects in software artifacts*, since design, code and test are produced based on the requirements informed by Organization B. *Defects in software artifacts* make Organization A to mobilize (and often overload) the development team to fix defects by performing *New urgent development activities*, which decrease the number of *Defects in software artifacts*. These urgent activities are performed as fast as possible, aiming not to delay other activities. Thus, they do not properly follow software quality good practices. Moreover, they contribute to increase the project cost and time (*Late and over budget project*). *Defects in software artifacts* increase the need of using *Software quality techniques* that, when used, lead to less *Defects in software artifacts*. This causal relationship has a delay, since the effect of using *Software quality techniques* can take a while to be perceived.

As shown in Figure 1, the archetype Shifting the Burden is composed of two balancing feedback loops and one reinforcing feedback loop. The balancing feedback loops (between *New urgent development activities* and *Defects in software artifacts*, and between *Defects in software artifacts* and *Software quality techniques*) mean that the involved variables influence each other in a balanced and stable way (e.g., higher/lower the number of *Defects in software artifacts*, more/less *New urgent development activities* are performed). In the reinforcing feedback loop, *New urgent development activities* are a symptomatic solution that leads to *Defects fixed through rework*, a

side effect, because once urgent development activities fix the defects in software artifacts, the Organization A feels like the problem was solved. This, in turn, decreases the need for using *Software quality techniques*, which is a more fundamental solution. As a result, software artifacts continue to be produced with defects, overloading the development team with new urgent development activities. Shifting the Burden is a complex behavior structure, because the balancing and reinforcing loops move the system (Organization A) in a direction (*New urgent development activities*) usually other than the one desired (*Software quality techniques*). *New urgent development activities* contribute to increase project cost and time (*Project is late and over budget*), because these activities were not initially planned in the project.

When Organization B does not properly define the project scope (*Scope poorly defined*), Organization A may allocate a *Team not suitable for the project*, contributing to *Defects in software artifacts* and also to *Changes in the project team* along the project. Usually, when the team is changed, the new members need to get knowledge about the project. Moreover, often the new members are more experienced and thus more expensive, which contributes to *Late and over budget project*. In order to change the project team, members can be moved from a project to another, causing *Deficit in other projects*. Furthermore, there is a balancing loop between *Changes in the project team* and *Defects in software artifacts*. The former may cause the latter due to instability inserted into the team. The latter, in turn, contributes to the former because *Defects in software artifacts* may lead to the need to change the team. There is a delay in this relationship because it can take a while between defects are noticed and the need to change the team. Finally, *Scope poorly defined* causes *Unrealistic deadline*, which contributes to *Late and over budget project*.

C. GUT Matrix. After getting a comprehensive view of the organization and how it behaves, we reflected about the behaviors on which the strategies should be focused. Thus, we created a GUT matrix to identify and prioritize behaviors of the system that are not fruitful, i.e., undesirable behaviors. They were identified mainly from the systemic maps. For example, from the fragment depicted in Figure 1, based on the positive causal relationship between *Misunderstood requirements* and *Defects in software artifacts*, the following undesirable behavior was identified: *Software artifacts are developed based on misunderstood requirements*. From the Shifting the Burden archetype, we identified: *Software quality techniques are not often applied to build software artifacts*. To complement information provided by the systemic maps, we used information from the interviews to look for behaviors the literature points out as desirable in organizations moving to agile (e.g., self-organized teams) [4].

After identifying the undesirable behaviors, the study participants validated and prioritized them considering the GUT dimensions. Each dimension was evaluated considering values from 1 (very low) to 5 (very high). 13 undesirable behaviors were identified. Table 1 shows a fragment of the GUT Matrix.

For each undesirable behavior, we analyzed the systemic maps and the interviews and identified its causes. (UB1) *Software artifacts are developed based on misunderstood requirement* because (C1) *Requirements are not satisfactorily described* and (C2) *Poor communication between client and development team*. C1 was identified directly from the systemic map. C2 was based on information about the procedure followed by Organization A to communicate with the client. When there is any doubt about requirements, the contact was made mainly through email or comments on issues in the project management system. Only Organization B has direct contact with the client.

Table 1: Matrix GUT fragment

#	Undesirable Behaviors	G	U	T	GxUxT
UB1	Software artifacts are developed based on misunderstood requirements	5	5	5	125
UB2	Software quality techniques are not often applied to build software artifacts	5	5	4	100
UB3	Projects are late and over budget	5	5	4	100
UB4	Organization has inconsistent knowledge of agile methods	5	5	4	100
UB5	Teams are not self-organized	5	4	4	80

C1 and C2 are also causes of (UB2) *Software quality techniques are not often applied to build software artifacts*, since the lack of well-defined requirements and direct contact with the client impact verification and validation activities. Moreover, there is a (C3) *Lack of clear and objective criteria to evaluate results* and (C4) *Large deliverables*, which make it difficult to evaluate results.

As it can be noticed in Figure 1, *Projects are late and over budget* (UB3) mainly because C1 and (C5) *Unstable scope and deadline*. Moreover (C6) *Unsuitable team allocation* and C4 also affect projects cost and time. The former because low productivity impacts on project time and, thus, cost. The latter because it is difficult to estimate large projects.

Regarding (UB4) *Organization has inconsistent knowledge of agile methods*, some members of the organization had previous experience with agile methods in other companies, others had a previous unsuccessful experience in Organization A and others did not have experienced agile methods. Most of the members were not sure about agile concepts and practices. Therefore, this undesirable behavior is caused by (C7) *Organization's members had different experiences with agile* and (C8) *Agile concepts and practices are not well-known by the organization*. Finally, *Teams are not self-organized* (UB5) due to the (C9) *Traditional development culture* that produced functional and hierarchical teams. Table 2 shows the identified causes and respective undesirable behaviors.

D.Strategies. The causes of undesirable behaviors and the prioritization made in the GUT matrix showed us leverage points of the system, i.e., points that if changed could change the system behavior. Therefore, we defined strategies to help Organization A move towards the second stage of StH by changing leverage points of the system and thus creating new behaviors in the system in that direction. We started by defining strategies to change undesirable behaviors at the top of the GUT matrix and causes related to more than one undesirable behavior.

Table 2: Causes of undesirable behaviors

#	Causes	UB1	UB2	UB3	UB4	UB5
C1	Requirements are not satisfactorily described	x	x	x		
C2	Poor communication between client and development team	x	x			
C3	Lack of clear and objective criteria to evaluate results		x			
C4	Large deliverables		x	x		
C5	Unstable scope and deadline			x		
C6	Unsuitable team allocation			x		
C7	Organization's members had different experiences with agile				x	
C8	Agile concepts and practices are not well-known by the organization				x	
C9	Traditional development culture					x

Considering Organization A characteristics, mainly its partnership with Organization B, the strategies combined agile and traditional practices. Agile approaches bring the culture of self-organized teams, shorter development cycles, user story, smaller deliverables, among other notions [4][8]. Traditional approaches were used to complement agile practices. After all, agile methods usually do not detail how to manage some aspects of a software project, such as costs and risks.

The first strategy (S1) consisted in establishing a new procedure to be followed by organizations A and B regarding requirements and communication, aiming to address C1 and C2. Due to business agreements, a big change in Organization B was not possible. For example, we could not change the fact that only Organization B could directly contact the project client. Hence, it was defined that requirements would be sent from Organization B to the project tech leader, who would rewrite the requirements as user stories and validate them with Organization B. By representing requirements as user stories, the project tech leader also has to represent their acceptance criteria, which aids to address C3. Moreover, to properly define the acceptance criteria, the tech leader needs to obtain detailed information about the requirement, stimulating Organization B to get such information from the client, which indirectly improves communication with the client. Only user stories defined according to the defined template and validated with Organization B follow to the next development activities. We also suggested the use of a template based on BDD (Behavior Driven Development) [22] describing business rules, acceptance criteria and scenarios to serve as a protocol to communicate requirements among organizations A, B and the client. It is worth mentioning that we were not allowed to ask Organization B to write the requirements itself by following the new guidelines, because this change was beyond the partnership agreements. In this strategy, we designated Organization B to play the Product Owner role. This way it is not only a business partner, but it represents the client interests and has responsibilities in this context. With this strategy, we also aimed to minimize the symptomatic solution (New urgent development activities) indicated in the Shift the Burden archetype identified in the systemic map. According to Meadows [11], the most effective strategy for dealing with a Shifting the Burden structure is to employ the symptomatic solution and develop the fundamental solution. Thus, it is possible to resolve the immediate problem and also work to ensure that it does not return. By improving requirements descriptions and defining clear acceptance criteria, software quality techniques (e.g., verification and validation), which are the fundamental solution identified in the Shifting the Burden, can be properly applied.

Another strategy (S2) focused on changing the undesirable behavior (UB3) *Projects are late and over budget*. Again, to change that, Organization A depended on changes in Organization B. Therefore, it was established that at the beginning of a project, Organization A and B should agree on the project scope, deadline, budget and involved risks. The project characteristics (e.g., technologies, domain of interest, platform, etc.) should also be clearly established. The project team would not be allocated before this agreement. By properly aligning information about the project between organizations A and B, it would be possible to allocate a development team with skills and maturity suitable for the project. By doing that, C5 and C6 would be minimized. Complementary, it was defined to change the development process as a whole. In the Organization A business model, when a project is contracted by a client, usually there is a cost and time associated to it. This prevented us from using a pure agile development process, were costs are dynamically established. As a strategy to implement tailored agile practices, it was defined: after requirements are validated, the development team (tech leader and developers) selects the requirements to be developed in a short cycle of development (i.e., a sprint), defines tasks and estimates time and costs related to them. This information is aligned between organizations A and B. This way, Organization B manages time and budget at project level, while Organization A manages time and budget in the sprint context. Once a week, monitoring meetings are performed to check time and budget performance. During the sprint, meetings based on the Scrum ceremonies are carried out in a flexible way. For example, if the team informs that there is nothing to report at the day, the daily meeting is not performed. Meetings that depend on client's feedback should be carried out with Organization B (in the Product Owner role). By breaking the development process in shorter cycles, C4 is addressed, since the product is also decomposed in smaller deliverables. This strategy also contributes to treat C9, as it changes the traditional development culture.

Aiming to change the way teams are organized in Organization A (UB5) and thus address C9, a strategy (S3) was defined to implement Squad and Guild [4] concepts. A Squad is a team with all skills and tools needed to develop and release a project. It is self-organized and can make decisions about its way of working. For example, a Squad can define the project development time-box (sprint) and how to implement some practices of strategies S1 and S2 (e.g., the use of BDD and how flexible Scrum ceremonies can be in the project). The members are responsible for creating and maintaining one or more projects. A Squad is composed of developers and a tech leader, who is responsible for communicating with Organization B mainly

regarding aspects related to budget, time and requirements. A Guild is a team responsible for defining standards and good practices that will be used for all squads. A Guild is composed of members with expertise in the subject of interest (e.g., a senior programmer can define good programming practices). Its purpose is to record and share good practices among the squads in the organization, aiming at achieving a homogeneous level of quality in the projects.

To address C7 and C8, which cause the organization to have inconsistent knowledge of agile methods (UB4), we defined as strategy (S4) the use of reference ontologies to provide a common conceptualization about the Software Engineering domain as a whole, and about the agile development process in particular. We used ontologies from SEON [16], a Software Engineering Ontology Network that describes various Software Engineering subdomains. Currently, it includes 18 ontologies related to subdomains such as Software Requirements, Software Project Management, Agile Software Process, among others. We extracted the view relevant to understand agile development. It contains a conceptual model fragment, axioms and textual descriptions that provide an integrated view of agile and traditional development, defining concepts in a clear, objective and unambiguous way. We suggested the use of SEON because its ontologies have been developed based on the literature and several standards, providing a consensual conceptualization. Moreover, we have successfully used it in several interoperability and knowledge-related initiatives. The SEON view used in the study includes concepts such as: Sprint, Deliverable, Project Deliverable, Software Requirement, User Story, Acceptance Criteria, Sprint Planned Task, Successfully Performed Task, among others.

Table 3 summarizes the defined strategies, the leverage points (causes) addressed by them and main agile concepts involved. It is worth noticing that some agile concepts were indirectly addressed. For example, although we did not directly use Product Backlog in S1, the set of requirements agreed with Organization B works as such. Similarly, in S3, when the team selects the requirements to be addressed in a development cycle, we are applying the Sprint Backlog notion. We decided not to use some of the original terms because Organization A had a bad previous experience trying to implement agile practices by following Scrum "by the book", which did not work and provoked resistance to certain practices. Thus, we tried to give some flexibility even to the practices' names, in order to avoid bad links with the previous experience.

After defining the strategies, they were executed by the organization in two projects with supervision of the first and third authors. The new practices started to be used in early February 2020. About four months later, we conducted an interview to obtain feedback. At that point, one of the projects had already been concluded and the other was ongoing.

Table 3: Strategies, Causes and Agile Concepts

#	Strategies	Agile Concepts	Causes
S1	New procedure to communicate requirements	User Story, BDD,Product Owner and Product Backlog	C1, C2, C3
S2	Budget and time globally and locally managed through short development cycles	Sprint, Sprint Backlog, Scrum meetings and Small deliverables	C4, C5, C6, C9
S3	Self-organized teams	Squad and Guild	C9
S4	Agile common conceptualization	Concepts related to agile software development	C7, C8

3.3 Study analysis, interpretation and lessons learned

In this section, we present results from the interviews that helped us to answer the research question, the resulting Systems Theory-based process that arose from this study and some lessons learned.

3.3.1 Results. To answer the research question, we carried out an interview with the software development director and the tech leader aiming to obtain their perception about the use of Systems Theory tools, GUT matrix and reference ontologies, as well as to get information about results obtained from the use of the defined strategies. They were interviewed together in a single section.

The director said that Systems Theory tools provided means to understand how different organizational aspects (e.g., business rules and quality software practices) are interrelated and influence each other, and how these aspects and interrelations produce desirable and undesirable behaviors. Moreover, Systems Theory helped to create strategies to change undesirable behaviors, since it provided a comprehensive understanding of the organization behavior and supported identifying causes of undesirable behaviors. Regarding GUT Matrix, the director stated that it was easy to use and important to prioritize the undesirable behaviors to be changed first. According to him, using these tools was easier and clearer when compared to Ishikawa and Pareto diagrams, because systemic maps allow more comprehensive and freer views and GUT matrix has a simple way of prioritization.

Concerning reference ontologies, he reported that they were useful to create a common communication among project stakeholders and business partners, eliminating some misunderstandings not only about agile practices but also about Software Engineering in general. For example, by using the conceptualization provided by the ontology, the team truly understood the "done" concept, commonly used in agile projects, in the sense that a software item (e.g., a functionality, a component) is done (i.e., ready to be delivered to the client) only if it met all the acceptance criteria established in the user stories materialized in that software item. The tech leader commented that by considering the ontology conceptualization, it was clearer the necessary information a requirement description should contain so that it can be properly understood. An interesting aspect pointed out by the interviewees was that the conceptualization provided by the reference ontologies was used by the development teams as a basis to quality rules in the projects (e.g., when a software item is done) and also to business rules in new business contracts (e.g., acceptance criteria need to be defined)

The director and tech leader informed that the first project in which the strategies were implemented was considered a successful

experience and served as a pilot. In similar projects, Organization A used to be 30% to 50% over time and budget due to spending extra resources on new urgent development activities to fix defects. By adopting the defined strategies, the project delivered a better product (at the moment of the interview, the client did not have reported any defect). However, the project was about 15% over budget and time due to changes in the agreed requirements. This may suggest that strategies S1 and S2 need adjustments. Although they seek to give some agility features to the development process, the project had its scope predefined by Organization B, which established it together with the client and set cost and time considering that scope. As Organization A started to develop the agreed requirements, Organization B noticed that some ot them needed to change to better satisfy the client needs. Although the project was late and over budget, the deviation in relation to the agreed cost and time was smaller than in similar projects that did not follow the strategies. The director pointed out that being able to show this difference to Organization B, indicating the causes that contribute to increase or decrease it, was an important result and can even be used to motivate Organization B to be more involved in the changes to improve the software development process as a whole. This would make it possible, for example, to adjust strategies S1 and S2 to make requirements elicitation, cost and time estimation more flexible.

The tech leader reported that using the strategies reduced misunderstandings in software requirements among the stakeholders and enabled better managing budget and time locally, in short development cycles. Moreover, according to him, in the second project adopting the strategies (ongoing project), the development team spent only 45 hours in new urgent development activities in a total of about 2000 hours of performed development activities. He also highlighted the use of user stories and BDD as an effective way to communicate requirements in this project.

The interviewees said that the self-organization culture has been developed in the teams and that the use of Squads has been very helpful. The use of Guilds was still in progress. Finally, they commented that, although the proposed strategies were used to address some undesirable behaviors by applying agile practices and concepts, they felt that changing the entire traditional culture can be a complex work, mainly because it requires to change mental models, processes and culture that also involve the organization partners (particularly Organization B) and clients.

3.3.2 Systems Theory-based Process. An important result that arose from this study is a process that combines Systems Theory tools and GUT Matrix to aid organizations move from traditional to agile. Figure 2 shows the process, and we briefly explain it next:

Understand the Organization: consists in obtaining information to understand the organization as a whole so that it will be possible to define strategies to implement agile practices in a suitable way for the organization, considering its culture, environment, business rules, software processes, agile experience and knowledge, people, and so on. Information can be obtained by using techniques such as interviews, document analysis and observation, among others.

Build a Systemic View: consists in using information obtained in the previous step to build systemic maps to understand organization behaviors relevant in the agile development context. Organization borders, relevant variables that drive organization behavior, causal relationships between them and feedback loops must be represented. Archetypes describing behavior patterns must also be identified from the systemic maps.

Identify Leverage Points: involves analyzing systematic maps and archetypes to identify undesirable behaviors and their causes. At this point, desirable behaviors in agile organizations suggested in the literature can also be used to verify if the organization fit them. Undesirable behaviors should be prioritized by using a GUT matrix, so that it is possible to identify which ones represent leverage points and will be addressed in the strategies.

Establish Strategies: consists in defining strategies (i.e., plans and actions) to implement agile practices focusing on the leverage points and considering the organization culture, business, rules, environment, people, etc.

Implement Strategies: involves implementing the defined strategies. It is suggested to start with one or two projects. After that, if the strategies work, they can be extended to other projects and then to the entire organization.

Monitor Strategies: consists in evaluating if undesirable behaviors changed as expected after strategies execution. The new behaviors caused by the strategies need to be evaluated and, depending on the results, strategies can be extended to other projects, aborted or adjusted.

3.3.3 Lessons Learned. In this section we discuss some lessons we learned in the study. In the lessons learned, we adopt terms such as *should* and *may* instead of mandatory terms such as *must* because we learned the lessons from a single case study. Thus, we believe that other studies are needed to corroborate what we have learned.

Systemic maps should be built with a goal in mind: since systemic maps allow to represent a comprehensive view of how the organization behaves and this may involve many aspects, it is important to focus on variables relevant to the goal to be achieved from the use of the systemic maps. Otherwise, the maps can be too complex and involve variables that do not provide meaningful information for the desired purpose.

The boundaries of the system should be clearly identified: to understand how external elements can influence organization behaviors, it is important to identify the organization boundaries as well the elements that the organization controls and the ones controlled by external agents. This way, it will be possible to create suitable strategies considering both the organization and the external agents.

Changes in leverage points may change the system as a whole: we noticed that when the changes are made in leverage points, particularly in the ones connected to undesirable behaviors with higher priority, the changes tend to provoke a meaningful shift in the organization behavior as a whole, changing existing behaviors and creating others. For example, by changing the way organizations A and B deal with project scope, time and budget, there were also changes in the way Organization A allocates teams, selects requirements to be implemented and the need for changes in the partnership rules with Organization B was perceived.

Strategies should be integrated into the software processes: in order for strategies to be performed as part of the organization daily activities, it is important that they are incorporated to the processes performed by the organization. In the study, the strategies were

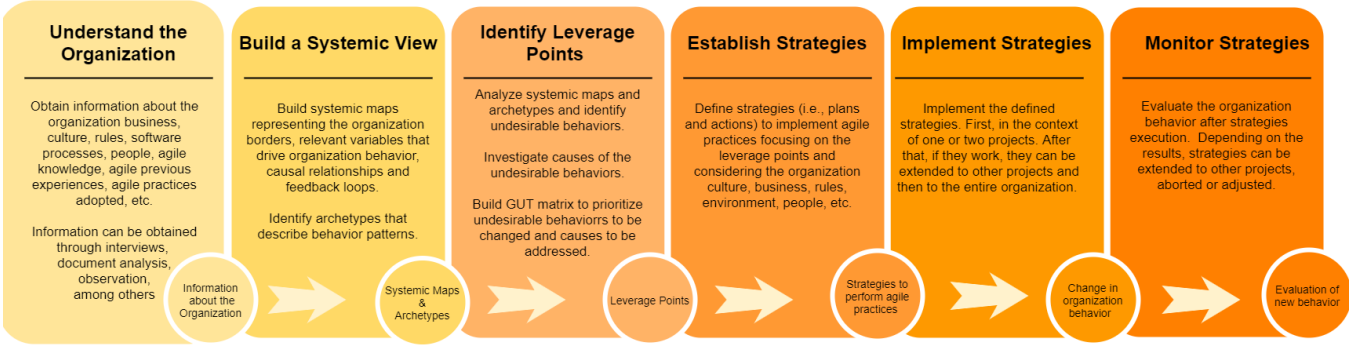

Figure 2: Process to aid defining strategies and implementing agile practices

incorporated into the organization software process, involving development, management and quality assurance activities.

Strategies should be gradually implemented and start in relevant projects: implementing the changes gradually and starting with one or two projects it was positive and the obtained results contributed for the organization to keep the intention of expanding the changes to other projects. We selected projects in which the teams were interested in using agile practices and that were important for the organization, so that the commitment of the team would be higher. This helped to minimize resistance to the new practices. Once they experienced the benefits of following the strategies, team members became disseminators of the new practices and concepts, helping to extend agile culture to other team members.

Strategies results should be measurable: when defining the strategies, we did not define any indicator to measure its effectiveness. However, the tech leaders used some metrics in the projects (e.g., number of hours spent in new urgent development activities, budget deviation, etc.) that helped us to evaluate the strategies. Thus, when defining the strategies, it is important to define the indicators to be used to evaluate them.

Representing the ontology conceptualization in textual format can be more palatable than conceptual models: the reference ontologies of SEON are represented by means of conceptual models, textual descriptions and axioms. Although the conceptual model of the SEON view used in the study provides an abstract view showing all the relevant concepts and relations in a single model, we noticed that the team preferred textual descriptions to the conceptual models. Thus, we prepared a document containing the concepts relevant to the study and their detailed description, also including information about constraints and relationships. This way, the conceptualization provided by the ontology was represented in a more palatable way for the team.

Changes involving business partners can be hard to implement and demand more flexibility and time: the way Organization B works directly affects Organization A. Due to business arrangements, Organization A does not have enough influence to make changes in Organization B. It can suggest changes, but it cannot demand them. Thus, it was necessary to define strategies that caused only small changes in Organization B (e.g., help to better describe requirements, allow shared control of time and cost). By noticing improvements from the use of the proposed strategies, Organization B may be more willing to further changes.

Squads should have autonomy to choose methods and tools: the organization can have a set of tools, techniques and methods to be adopted in the projects. Guilds can help define that. According to the project team and characteristics, some tools, methods and techniques can suit better. We noticed that the squad became more self-organized when its members could choose the techniques to solve the project problems. For example, in the study, a squad decided to adopt user stories and BDD to describe requirements, while the other used the complete user story template. In both cases, information about requirements was clear and complete. However, each squad chose the technique more suitable for the project and team characteristics.

4 THREATS TO VALIDITY TO THE STUDY RESULTS

The validity of a study denotes the trustworthiness of the results. Every study has threats that should be addressed as much as possible and considered together with the results. In this section, we discuss some threats considering the classification proposed in [15].

The main threat in this study is related to the researchers who conducted the study. Participative case studies are biased and subjective as their results rely on the researchers [1]. The first and third authors acted as consultants in Organization A and were responsible for conducting the interviews, creating systemic maps and GUT matrix and defining strategies. The researchers participation affects *Internal Validity*, which is concerned with the relationship between results and the applied treatment, *External Validity*, which regards to what extent it is possible to generalize the results from the case-specific findings to different cases, and *Reliability Validity*, which refers to what extent data and analysis depend on specific researchers. To reduce the threat, the other study participants participated in the activities and validated results. Moreover, another researcher, external to the organization, evaluated data collection and analysis and was involved in discussing and reflecting on the study and results.

Concerning *Construct Validity*, which is related to the constructs involved in the study, the main threat is that we did not define indicators to evaluate results. Data collection was performed through interviews, which involves subjectivity. To minimize this threat,

we used some measures collected in the projects to evaluate the new behaviors caused by the proposed strategies.

In case-based research, after getting results from specific case studies, generalization can be established for similar cases. However, the threats aforementioned constraint generalization. Moreover, the study involved only one organization. Thus, it is not possible to generalize results for cases without researcher intervention or for organizations not similar to Organization A.

5 CONCLUSIONS AND FUTURE WORKS

This paper presented a case study carried out in a Brazilian organization towards the first transition in the path prescribed by the StH model [12]. Organization A develops software in partnership with a European organization (Organization B) and it does not have direct contact with clients. After went through an unsuccessful try to implement agile practices "by the book", the organization started a long-term process improvement program. To support it, we have used StH to describe the evolution path to be followed. To aid in the first transition and move from traditional to agile, we combined Systems Theory tools, GUT matrix and reference ontologies.

In summary, Systems Theory tools and GUT Matrix were helpful to better understand the organization, find leverage points of change and define strategies aligned to the organization characteristics and priorities. Reference ontologies were useful to establish a common understanding about agile methods, enabling teams to be aware of and, thus, more committed to agile practices and concepts.

As a result of the initiative, the organization has implemented agile practices in a flexible way and combined to some traditional practices, which is more suitable for the organization characteristics. Due to the obtained results, the organization kept its intention to continue evolving by following the StH stages. In the first transition, it was not possible to propose big changes in the way Organization B works. However, Organization A expects that considering the positive results, Organization B will be more willing to be involved in the evolution path. This will be crucial in the more advanced stage, where data from the clients are needed to support decision-making and identify new opportunities.

Regarding human aspects, we focused mainly on soft skills related to agile culture. Strategy S3 is directly related to human aspects, being responsible for implementing Squads and Guilds. Squads promoted self-organization, trust, leadership and others important skills in agile organizations. Guilds promoted creation of processes and organizational culture that enabled sharing and managing knowledge at individual, team and organizational levels. This knowledge is valuable to continuous improvement of Organization A. By changing human aspects, S3 enabled Organization A to create processes, vocabulary and mindset, i.e., an organizational culture that supported the movement from traditional to agile. Moreover, the soft skills developed by S3 supported other strategies. For example, S1 and S2 were possible because S3 developed some soft skills (e.g., effective communication, self-organization and adaptability) that supported S1 and S2.

As for limitations of our approach, we highlight that it involves a lot of tacit knowledge and judgment. Besides knowledge about System Thinking tools and GUT matrix, it is necessary to have organizational knowledge to apply them (e.g., one must be able to properly identify problems, investigate causes, define strategies etc.). As future work, we plan to add knowledge (e.g., by means of guidelines) to help others to use our approach. We also intend to explore other Systems Theory tools and combine them with Enterprise Architecture Models to connect system variables, undesirable behaviors and causes to elements of the organization architecture. Concerning Organization A, we plan to monitor the implemented strategies and extend them to other projects. Once the new practices become solid, we plan to aid Organization A in the next transitions, where continuous integration and continuous deployment are performed.

REFERENCES

[1] Richard L Baskerville. 1997. Distinguishing action research from participative case studies. *Journal of systems and information technology* 1, 1 (1997), 25–45.

[2] Jan Bosch. 2014. Continuous Software Engineering: An Introduction. In *Continuous Software Engineering*. Springer, Chapter 1, 3–13.

[3] Thatiany Lima De Sousa, Elaine Venson, Rejane Maria Da Costa Figueiredo, Ricardo Ajax Kosloski, and Luiz Carlos Miyadaira Ribeiro. 2016. Using scrum in outsourced government projects: An action research. In *2016 49th Hawaii International Conference on System Sciences (HICSS)*. IEEE, 5447–5456.

[4] Leffingwell Dean. 2016. SAFe® 4.0 Reference Guide: Scaled Agile Framework® for Lean Software and Systems Engineering.

[5] Tore Dybå and Torgeir Dingsøyr. 2008. Empirical studies of agile software development: A systematic review. *Information and Software Technology* 50, 9-10 (2008), 833–859.

[6] Peggy Gregory, Leonor Barroca, Helen Sharp, Advait Deshpande, and Katie Taylor. 2016. The challenges that challenge: Engaging with agile practitioners' concerns. *Information and Software Technology* 77 (2016), 92–104.

[7] Giancarlo Guizzardi. 2007. On ontology, ontologies, conceptualizations, modeling languages, and (meta) models. *Frontiers in Artificial Intelligence and Applications* 155 (2007), 18–39.

[8] Teemu Karvonen, Lucy E. Lwakatare, Tanja Sauvola, Jan Bosch, Helena H. Olsson, Pasi Kuvaja, and Markku Oivo. 2015. Hitting the target: practices for moving toward innovation experiment systems. In *Int. Conference on Software Business*. 117–131.

[9] Charles Higgins Kepner and Benjamin B Tregoe. 1981. *The new rational manager*. Vol. 37. Princeton Research Press Princeton, NJ.

[10] Daniel H Kim. 1993. *Systems archetypes I: diagnosing systemic issues and designing high-leverage interventions*. Pegasus Communications.

[11] Donella H Meadows. 2008. *Thinking in systems: A primer*. Chelsea Green Publishing.

[12] Helena Holmstrom Olsson, Hiva Alahyari, and Jan Bosch. 2012. Climbing the "Stairway to Heaven" –A Mulitiple-Case Study Exploring Barriers in the Transition from Agile Development towards Continuous Deployment of Software. In *38th Euromicro Conf.on Software Engineering and Advanced Applications*. 392–399.

[13] Helena H. Olsson and Jan Bosch. 2014. Climbing the "Stairway to Heaven": evolving from agile development to continuous deployment of software. In *Continuous Software Engineering*. Springer, Chapter 10, 15–27.

[14] Pilar Rodriguez, Jouni Markkula, Markku Oivo, and Kimmo Turula. 2012. Survey on agile and lean usage in finnish software industry. In *ACM-IEEE Int. Symposium on Empirical Software Engineering and Measurement*. 139–148.

[15] Per Runeson, Martin Host, Austen Rainer, and Bjorn Regnell. 2012. *Case study research in software engineering: Guidelines and examples*. John Wiley & Sons.

[16] Fabiano B. Ruy, Ricardo A. Falbo, Monalessa P. Barcellos, Simone D. Costa, and Giancarlo Guizzardi. 2016. SEON: a software engineering ontology network. In *European Knowledge Acquisition Workshop*. Springer, 527–542.

[17] John Sterman. 2010. *Business Dynamics*. Irwin/McGraw-Hill.

[18] John D Sterman. 1994. Learning in and about complex systems. *System Dynamics Review* 10, 2-3 (1994), 291–330.

[19] Rudi Studer, V Richard Benjamins, and Dieter Fensel. 1998. Knowledge engineering: principles and methods. *Data & knowledge Engineering* 25, 1-2 (1998), 161–197.

[20] Richard Vidgen and Xiaofeng Wang. 2009. Coevolving systems and the organization of agile software development. *Information Systems Research* 20, 3 (2009), 355–376.

[21] Laurie Williams and Alistair Cockburn. 2003. Agile software development: it's about feedback and change. *IEEE Computer* 36, 6 (2003), 39–43.

[22] Matt Wynne, Aslak Hellesoy, and Steve Tooke. 2017. *The cucumber book: behaviour-driven development for testers and developers*. Pragmatic Bookshelf.