A Requirements Elicitation and Analysis Approach in SAFe model For Enterprise Companies

Hessam Emami
Southern Methodist university

Abstract

Software Requirement Engineering consists of different stages. From requirement elicitation and analysis to verification and validation, there is a wide variety of strategies and techniques that are used proportional to company size, business model, goal, mission, and other factors. Requirement elicitation and requirement analysis due to complicated business and technical dependencies in large companies could become a daily challenge for stakeholders. This paper study the effectiveness of common Requirement Engineering approaches for requirement elicitation and analysis in SAFe model and presents a proposed approach to mitigate the obstacles.

Keywords- Requirement elicitation; Requirement analysis; Software Engineering

1. Introduction

No matter if large enterprise companies dedicate a considerable part of their resources to establish IT departments, more rely on third-party software companies or both, they will not be able to escape from managing their requirements. Without proper requirement engineering practices, it is unlikely that stakeholder decisions lead to desired results. Requirement elicitation and requirement analysis are two stages in requirement engineering that have a high impact on the outcome.

Established companies own mature process, talent pool, maturity, partners and a lot more while startups lack all these capabilities initially [2]. Therefore, this assumption might exist that by following established practices, success is secured. Numerous studies have shown there is a high percentage of project failure exits in large companies primarily linked to the shortage of proper upfront planning.

Although the Waterfall model still is popular, many companies have been striving to encourage their project managers to employ the Agile mindset in their project management model. Scaled agile framework (SAFe) is the world's leading framework for scaling Agile across the enterprise [7]. Used by hundreds of the world's largest organizations, SAFe sustains and drives faster time-to-market, dramatic increases in productivity and quality, and improvement in employee engagement [7].

The goal of this paper is to investigate the practicality of requirement elicitation and analysis procedures in SAFe model for enterprises and propose a solution that can reduce the chance of project failures due to requirement ambiguities.

In section 3, challenges are discussed. In section 4, finding from section 3 and other studies have been used to demonstrate potential solutions to improve requirement elicitation and analysis practices in SAFe model. In section 5, a proposed tool is discussed to facilitate the process in SAFe model. In section 6, a use case is study is considered to demonstrate the practicality of the approached.

2. Background

Scaled agile framework attempts to incorporate the various lessons from Lean and Agile methodologies into the basic principles, which are then used to bring substantial improvements to time to market, employee engagement, quality, and productivity [8]. SAFe promotes Lean and Agile practices into traditional corporate organizations [8]. Stakeholders, project managers, and development teams work together at different levels to elicit, analyze, implement, release, and verify the requirements. Every company implements SAFe uniquely; however, in most cases, they follow its principals, therefore, to examine elicitation and analysis techniques, the article presents the following empirical implementation of SAFe, which

is close to the majority of studied enterprises in this research.

In this practical implementation of SAFe, the highest level in Agile Release Train (ART), portfolio level, includes executive manages who provide goals for the future Program Increments (PI). At the Program level, delivery managers, architects, and system engineers work with design, project managers, and Experience Owners (EOs) to create requirements at Epic level. Eventually, IT Scrum teams work with their project managers to break down Epics to Features. This one of the possible approaches to utilize SAFe in an enterprise and the purpose of this study is not to research on SAFe methodology but using the mentioned implementation as a baseline to study requirements elicitation and analysis in SAFe.

3. Problems

The process of requirement elicitation in our SAFe models starts by defining goals by executive managers. These goals are aligned with enterprise goals and business models. Experience Owners break down the goal into sub goal, and as each Line of Business (LOB) consists of multiple Applications defining sub goals for each Application can be complicated. Also, it is difficult to find out in advance all sub goals are aligned with the goals. The following table demonstrates the relationship between goals and sub goals defined in one of the studied companies.

Table 3.1 goal and sub goals

Improve company reputation in states x and y	Improve SEO in the website
	Increase activities in social media
	Swath to third-party x for y services.

Although table 3.1 clearly shows what executive managers and Experience Owners see in their business road map, it does not demonstrate how sub goals can fulfill the goal.

Architects, designers, and EOs develop Epics from sub goals. Providing high-level requirements by architects and designers that can fulfill Epic objectives is challenging. In enterprise companies, it is common that solutions have dependencies or overlaps with each other. Reusability of current systems and Integration with other platforms is the highest priority for architects. If these systems are not integrated properly, further consequences may be poor business process and unforeseen costs, then reconciling the data [4] on the hand design teams tend to provide a better user experience and does not make compromises. Creating features from Epics in many cases shows high-level designs that are supplied by architects are not feasible to be implemented. Architects working at a different level causes being detached from the implementation and their requirement analysis not mapped with technologies that are used in the projects. It is clear that SAFe can bring some advantages of Agile and linear models. Still, by defining different levels and lengthy incremental delivery, it can cause issues regarding requirement elicitation and analysis in complex projects.

In a case study on one of the Scrum team in a large company utilizing SAFe the result 3.2 was found:

Table 3.2 Requirement elicitation

Goal	Avg Time Define Sub goal	Avg Time Deliver Reqs To Dev Teams	Number Of Changed Reqs	Total Req
1	2 months	4 months	15	17
2	2 months	4 months	3	13
3	6 months	1 months	2	8

In this study, it is shown on average, 50 percent of the requirements have to be changed after delivery to the development team. In another study on the same team with the same goal (table 3.3) we can observe after two years of development, all projects failed to be delivered to production.

Table 3.3 goal to production

Goals	Development Time	Delivered to users
1	18 months	No
2	2 months	No
3	6 months	No

4. Proposed Approach

In our SAFe model, goals are defined at the executive level. This study approach focusses on eliciting requirements and analyzing them from sub goal level and below. The root cause of problems discussed in the previous section can be categorized into the following groups:

- Business Involvements
- Domain knowledge
- Software Integration

This approach presents a process to address the problems by focusing on root causes by applying other studies in this field.

As the complexity of software system increases requirements elicitation and analysis are becoming increasingly difficult in software development [1]. In paper [1], rules are determined for requirements elicitation, and a process is described for requirements analysis. We will enhance the process by applying the following changes:

1. Requirements elicitation and analysis in each level
2. Iterative process
3. Feedback loops

In SAFe model, stakeholders at each level have defined responsibilities. It is easy to make this premise that EOs and project managers are the only contributors who should manage the requirements. If there are different levels of requirements, different procedures are also required to achieve the desired outcome. For the first attempt, we defined the following rules by getting help from study [1] for requirement elicitations and proposed the process [1] for each level:

- <Who> is responsible to input <what> into the system and <how> [1]
- <What> to <Who> and <How> system generates output.
- <Why> output needs to be generated.

The "Why" questions are the key to map sub goals to goals in the elicitation stage.

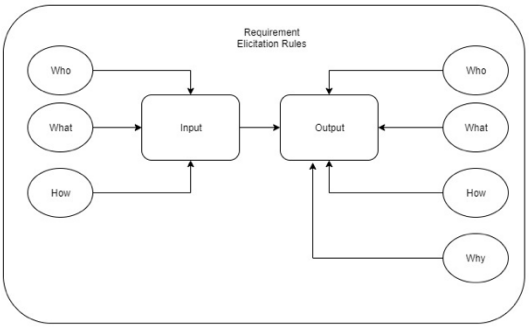

Figure 4.1 input output and rules relationship

Although all these rules can be used in all levels for requirements elicitation, requirements process will be different at each level. In addition, a feedback loop needs to be considered in order to other members in ART to be notified about the quality of the defined requirements at their level. As feedbacks might challenge the requirements, an iterative approach can refine the requirements based on analysis generated by lower levels. Designers and architects have essential roles in program level to deliver a high-level design that can be feasible to implement, able to be integrated with other systems, and provides the desired user experience. It is EOs and project manager responsibilities to define the sub goals and work with IT and design to write Epics from them. Figure 4.2 shows the analysis process at this level:

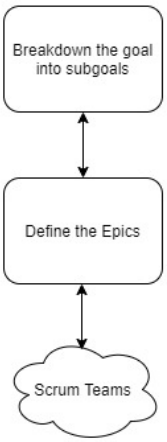

Figure 4.2 Program level process

The most significant advantage of this process compares to [1] process is the internal feedback at this level. After EOs and project managers defined the sub goals and presented to architects and designers, it frequently happens that in the middle of requirement analysis, sub goals need to be broken down into small goals. One of the factors that seem to influence an individual's effectiveness in requirements engineering activities is knowledge of the problem being solved, i.e., domain knowledge [5]. Contributors at the program level might have a different understanding of the problem result from a different perspective, so an iterative approach can facilitate to include all angles.

In the next step, Scrum teams and project managers define the feature from the Epics:

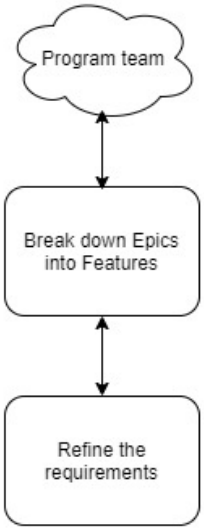

Figure 4.3 Development team level process

There are two architectural modes of integrating systems. [4] The first is to integrate all the systems directly with each other, i.e., one-to-one. [4]. The other mode of integration is to develop an intermediator (integration bus) to which all systems are connected. [4] Large organizations mostly support both cases, and that makes it tough for an architect to examine the best solution for high-level design. The design team also might not able to perform A/B testing or another usability experiment at the time of requirement analysis. It is essential that Scrum teams also offer their feedback about the Epics and request Epic refinement from the Program team as they have hands-on experience on technologies and sometimes access to users more than the program team.

4.2. Epics to Feature Stages in more details

Among all the steps defined in this section, Epics to Features demands to be explored more. This step is placed at the program level. In opposite to Scrum teams that are mostly include IT personals, the program team involves business, IT, and design teams. Requirement elicitation and analysis is a shared responsibility across all the team members, but the type of accountability is different in these teams. Collaboration among teams at the program level to elicit and analyze the requirements entails a subprocess in order to break down the sub goals to Epics.

The figure 4.4 shows this sub process at the program level:

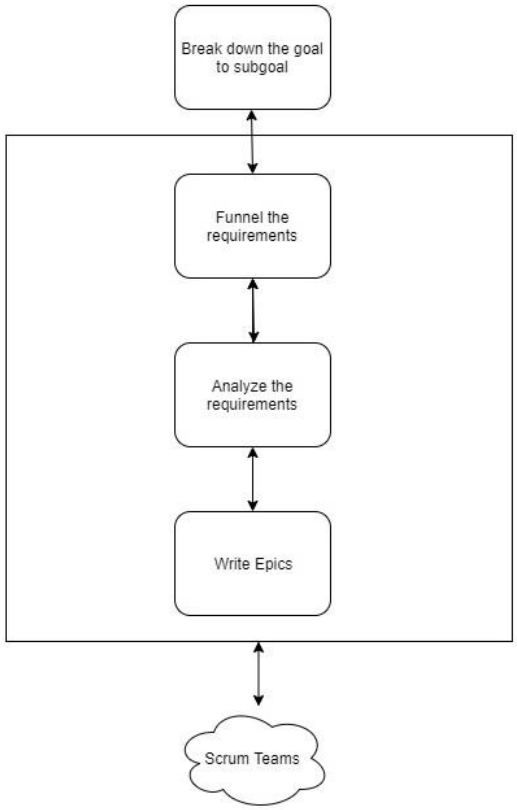

Figure 4.4 Program level detailed process

As it is shown, architects and designers do not work on sub goals directly. Requirements determined by EOs from sub goals will be collected and pass into Funnel state. At this state, EOs attempt to discover overlaps or contradiction between requirements and sub goals. In the next step, architects and designers analyze the requirements, and if no refinement is required, Epics will be generated in the last step.

5. Proposed Tool

Simple recording methods such as test, lists, sketches, etc. appear to be the accepted practice in recording requirements knowledge [6]. These recording methods are unfeasible or unhelpful in the long run because of essential difficulty in organizing and using the produced records [6].

The purpose of this tool is to give an ability to stakeholders to document the requirements by using specified rules, establish the relationship between requirements and search among them. The proposed tool provides an online GUI which is accessible through different platforms. This tool can be extended to cover another stage of requirement engineering, but the focus of this reading is only restricted to elicitation and analysis stages. There are four types of requirement can be tacked in the application:

- Goal
- Sub Goal
- Epic
- Feature

Each of these types have shared and unique attributes

Table 5.1 Goal rule attributes

Goal	Stakeholders
	Description

Goal type value is controlled by executive mangers and in this paper, it is assumed a predefined goal is available for the ART.

Table 5.2 Sub goal rule attributes

Sub goal	Stakeholders
	Description
	Input/Who
	Input/What
	Input/How
	Output/Who
	Output/What
	Output/How
	Output/Why
	Step
	Parent goal

Each rule that is defined in the approach needs to be stored for the sub goal. In addition, as generating Epics from sub goal has different steps, step attribute is stored as well. Each goal can be related to one parent goal, but each parent goal can have multiple goals.

Table 5.3 Epic rule attributes

Epic	Stakeholders
	Description
	Input/Who
	Input/What
	Input/How
	Output/Who
	Output/What
	Output/How
	Output/Why
	Accepted by Scrum team
	Parent sub goal

Accepted by Scrum team attribute means Epic can be broken down into features. Each Epic can be only related to on sub goal, but each sub goal can have multiple Epics.

Table 5.4 Feature rule attributes

Feature	Stakeholders
	Description
	Input/Who
	Input/What
	Input/How
	Output/Who
	Output/What
	Output/How
	Output/Why
	Parent Epic

Feature is the last phase of requirement elicitation and analysis in our approach and does not need any approval from the other teams. Each feature can only be related to one Epic, but each Epic can have multiple features.

The infrastructure for this tool supports real-time updates, notification systems, and roles. Goals can only be defined and modified in the system by executive managers, however, can be read by Experience Owners.

Sub goal can be modified by experience owners but can be read by all other stakeholders. Epic will be written by architects, project managers, and designers and can be read by all stakeholders. Features is controlled by Scrum teams but can be read by all stakeholders. Authorization access mentioned above are the default settings and can be modified in the system by the system admin. Figure 5.1 demonstrates the proposal platform for this tool. This system will detect any incomplete or redundancy in data that is against the rules. GUI, by providing the ability to create and modify requirement types can satisfy technical and nontechnical stakeholders.

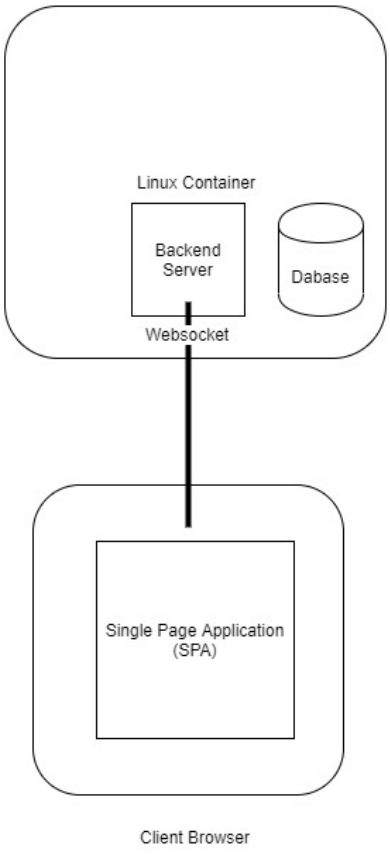

Figure 5.1 Tool HDL

6. Case Study

In one of the studied enterprise companies, we requested to get access to one of the next PI goals and let us to elicit and analyze the requirements by the proposed approach. It requires to be mentioned it was not possible to access all the program team and Scrum members, so some refinements are required for the final requirements. Also, the outcome of this study could not be collected at Scrum level at the time of writing the paper as Program Increment in the studied company takes three months, and Scrum teams were in the middle of their PIs.

The following is the studied goal for the ART:

"Increase product x sell by y percent by end of 2020".

Requirement elicitation that the company uses mainly relies on analytics data. Experience Owners have a close collaboration with their data analytics team to find the major pain point of their customers. Experience Owners found out the process that gives the customers the purchasing price of the service x is lengthy and complicated. Therefore, they assumed reducing the number of steps to get access to the price will help to encourage more users to buy the product x. Also, they learn that the majority of their customers leave the page after they are asked to provide some personal information. To resolve the problem, they decided to move the personal information page form the middle of the flow to the last step when the customer is ready to sign up and pay for the service. Provided tables omitted stakeholder information. EOs defines the following sub goals at this step:

Table 6.1 Case study sub goals

Sub goal 1
Increase customers confidence in the system during registration flow for product x
Input/Who Product x cusotmers
Input/What Remove Personal information page
Input/How Move the Personal information page to then end of the flow
Output/Who Customers
Output/What New flow for product X
Output/How Replace the current flow
Output/Why Reduce the registration time for product x
Step Analyze
Goal 1

After this step, it was asked from program team to write Epics that can fulfill the requirements. Not all collaborators in program team participated in this effort.

Table 6.2 Case study Epics1 goals

Epic 1
Update REST services for product x flow
Input/Who Product x customers
Input/What Customer REST API in project x
Input/How Develop new controllers to support pilot testing
Output/Who Product x customers
Output/What Access to new end points
Output/How Release new REST APIs
Output/Why Provide ability for UI to call new REST end points and ability to run pilot testing
Pending acceptation from Scrum teams
Sub goal 1

Table 6.3 Case study Epics 2

Epic 2
Update UI to support new product x flow
Input/Who Product x customers
Input/What Update SPA react application
Input/How Update routing handler in product x SAP and Axios endpoints
Output/Who Product x customers
Output/What Ability to utilize new API
Output/How Using new flow in production
Output/Why Provide ability for UI to call new REST end points and ability to run pilot testing
Pending acceptation from Scrum teams
Sub goal 1

It was observed that this process served both EOs and IT personnel's in the program team to speak with each other in different languages. Scrum teams did not participate in this study. No features have been created out of these Epics at the time of writing this study, however, the table 6.4 shows how the process reduced the time without compensation on any requirement elicitation and analysis process.

Note: *This study did not follow all industrial and academic practices for data collection. For this type of study, one recommended approach is A/B testing that uses two groups of personnel and compares the timing between the group that uses the new method against the group that uses the traditional one. Presenting the article had a deadline, and the author could not perform the mentioned tests.*

Table 6.4 Case study results

Goal	Avg Time Define Sub goal	Avg Time Deliver Reqs To Dev Teams
1	4 days	6 days

Table 6.4 shows a 450 percent time reduction in sub goal definition and 360 percent improvement in requirements delivery time. Although the result is promising, these requirements are not delivered to Scrum teams for evaluation. It is highly probable that Scrum teams reject some of the suggestions provided at the Epics and ask for refinements of the requirements.

7. Conclusions

SAFe model is getting attention in medium and large companies. What SAFe claims is it can deliver Agile mindset but also some level of control for the management team. The biggest challenge for implementing Agile in Enterprise is executive managers and delivery managers don't communicate with each other frequently, so a mechanism should be in place to feel this gap. Agile Release Train delegates some of the responsibilities from Scrum teams to the program level. IT and business members in program teams work as an interface between the portfolio team and Scrum teams. By running Program Increment ceremonies, executive managers would have the chance to measure the alignment of Experience Owner works against the business roadmap. Each PI has specific aims usually defined by sub goals, and Scrum teams are the development team to deliver the project. This structure suffers a significant concern. Team members who make high-level decisions about the requirements are detached from the groundwork. They might not be aware of the hardness of the task that it is demanded, actual dependencies, effort, and complexity of the project.

In many cases it is shown that requested requirements do not even need to be implemented. In this study, the goal approach for requirement elicitation and analysis proposed by [1] is revisited. The paper's method emphasizes that requirements management is the responsibility of all stakeholders. The type and level of the effort might be different, but business and system requirements cannot be collected in the silo. An iterative approach is explicitly required at program level as the stakeholders at that level are not in the same groups. Also, Scrum teams should provide their feedback to the program level due to familiarity with the technologies.

Rules in this method work as checkpoints in order to guide stakeholders in eliciting and analyzing the requirements. The process in the place can demonstrate the status of the requirements and notify the stakeholders at the same or different levels about the potential obstacles. In the end, the designed tool can improve stakeholder's performance and replaces legacy techniques that might not easy to use and manage the requirements down the road.

References

[1] M. Gopi Chand, K. Narendar Reddy, A. Ananda Rao and J. Kiran Kumar, "An approach to requirements elicitation and analysis using goal," 2010 2nd International Conference on Software Technology and Engineering, San Juan, PR, 2010, pp. V1-218-V1-221.

[2] U. Rafiq, S. S. Bajwa, X. Wang and I. Lunesu, "Requirements Elicitation Techniques Applied in Software Startups," 2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA), Vienna, 2017, pp. 141-144.

[3] I. Basharat, M. Fatima, R. Nisa, R. Hashim and A. Khanum, "Requirements engineering practices in small and medium software companies: An empirical study," 2013 Science and Information Conference, London, 2013, pp. 218-222.

[4] H. J. Rognerud and J. E. Hannay, "Challenges in enterprise software integration: An industrial study using repertory grids," 2009 3rd International Symposium on Empirical Software Engineering and Measurement, Lake Buena Vista, FL, 2009, pp. 11-22.

[5] A. Niknafs and D. M. Berry, "The impact of domain knowledge on the effectiveness of requirements idea generation during requirements elicitation," 2012 20th IEEE International Requirements Engineering Conference (RE), Chicago, IL, 2012, pp. 181-190.

[6] K. Yozgyur, "A proposal for a requirements elicitation tool to increase stakeholder involvement," 2014 IEEE 5th International Conference on Software Engineering and Service Science, Beijing, 2014, pp. 145-148.

[7] The Provider of SAFe. (n.d.). Retrieved from https://www.scaledagile.com/.

[8] Portfolio Level. (n.d.). Retrieved from https://www.scaledagileframework.com/portfolio-level/.

[9] H. Dar, M. I. Lali, H. Ashraf, M. Ramzan, T. Amjad and B. Shahzad, "A Systematic Study on Software Requirements Elicitation Techniques and its Challenges in Mobile Application Development," in IEEE Access, vol. 6, pp. 63859-63867, 2018.

[10] A. Knauss, "On the usage of context for requirements elicitation: End-user involvement in IT ecosystems," 2012 20th IEEE International Requirements Engineering Conference (RE), Chicago, IL, 2012, pp. 345-348.

[11] O. Dieste, N. Juristo and F. Shull, "Understanding the Customer: What Do We Know about Requirements Elicitation?," in IEEE Software, vol. 25, no. 2, pp. 11-13, March-April 2008.