2008

Requirements elicitation using goal-based organizational model

Aneesh Krishna

University of Wollongong, aneesh@uow.edu.au

He Lu

University of Wollongong

Follow this and additional works at: https://ro.uow.edu.au/infopapers

Part of the Physical Sciences and Mathematics Commons

Recommended Citation
https://ro.uow.edu.au/infopapers/3224

Research Online is the open access institutional repository for the University of Wollongong. For further information contact the UOW Library: research-pubs@uow.edu.au
Requirements elicitation using goal-based organizational model

Abstract
Domain knowledge is crucial for the quality of requirements. The systems analysts, who have adequate knowledge of software often don't understand the organizational context well. The Tropos methodology proposes a set of concepts, which are effective in capturing the organizational context, the interactions between the software system and the human agents. Based on the Tropos methodology, Software System-Business Model (SS-BM), which integrates the Software System Actor (SSA) from early requirements organizational model has been proposed. In this paper, a methodological approach of applying the SS-BM to requirements list elicitation is presented. In the proposed method, requirements items were mapped to SS-BM elements based on their semantic meanings. The mapped requirements list and SS-BM were analyzed in a specific sequence according to multiple checking points. The problems of requirements list were identified in the analysis process. For each kind of problem identified a possible improvement is proposed. Quality indicators were used to summarize the quality of requirements at the end of the elicitation process.

Disciplines
Physical Sciences and Mathematics

Publication Details

This conference paper is available at Research Online: https://ro.uow.edu.au/infopapers/3224
Requirements Elicitation using Goal-based Organizational Model

Aneesh Krishna, He Lu
Decision Systems Laboratory
School of Computer Science and Software Engineering
University of Wollongong, NSW 2522, Australia
{aneesh, hl}@uow.edu.au

Abstract

Domain knowledge is crucial for the quality of requirements. The systems analysts, who have adequate knowledge of software often don’t understand the organizational context well. The Tropos methodology proposes a set of concepts, which are effective in capturing the organizational context, the interactions between the software system and the human agents. Based on the Tropos methodology, Software System-Business Model (SS-BM), which integrates the Software System Actor (SSA) from early requirements organizational model has been proposed. In this paper, a methodological approach of applying the SS-BM to requirements list elicitation is presented. In the proposed method, requirements items were mapped to SS-BM elements based on their semantic meanings. The mapped requirements list and SS-BM were analyzed in a specific sequence according to multiple checking points. The problems of requirements list were identified in the analysis process. For each kind of problem identified a possible improvement is proposed. Quality indicators were used to summarize the quality of requirements at the end of the elicitation process.

1. Introduction

A systems analyst is responsible for collecting the business needs from their stakeholders and generating requirements specification. This is done on the basis of information provided by the clients in written format. Requirements elicitation is one of the most crucial steps in the software development process. Although system analysts have adequate knowledge of the software technology, they often don’t understand well, the organizational context in which the future system will be situated [6]. System analysts’ lack of domain knowledge may result in poor requirements elicitation and even the failure of the software project.

Tropos [6][7] is a methodology, which is intended to support five phases of software development: early requirements, late requirements, architectural design, detailed design and implementation. In recent years, considerable research efforts have been made on both early requirements [5][9] and the late requirements [8][9]. In early requirements phase, analysts study the existing organization to understand the problem. The output of early requirements is an organizational model [10], which includes relevant actors, their goals and dependencies. In late requirements phase, a FELRE (From Early Requirements to Late Requirements) pattern language[1] is used to transform the organizational model to a new Software System-Business Model (SS-BM), which integrates the Software System Actor (SSA) and expresses the functionality of the software system[1][2].

We propose a method which uses SS-BM to evaluate and elicit requirements from the requirements list. Requirements list is a document provided by the customer to describe business needs at the beginning of the project, which is written in natural language. The SS-BM evolving from the organizational model expresses the functionality of the software system in a semi-formal language. In our method, we make use of SS-BM (which is based on the Tropos methodology) to evaluate the requirements list, then the requirements list can be optimized to reflect the intentions of the stakeholders more accurately. In the mean time, the risk generated by the systems analysts’ lack of domain knowledge is decreased considerably. After this optimization process, the prioritized requirements list can be an improved base for the requirements specification. Our proposed method allows systems analyst to detect the correctness, completeness and hierarchical consistency of the requirements list and suggest improvement in the requirements elicitation process. Firstly, the SS-BM elements and requirements items are connected by map items in a systematic way. Then, the mapped requirements list and SS-BM were scanned in a specific sequence. Each requirements item and SS-BM element were analyzed according to multiple checking points. Quality indicators are then used to identify the quality of requirements. Proposed requirements elicitation process is based on the correctness of SS-BM hypothesis. This paper is structured as follow: Section 2 presents an overview of the proposal. Section 3 introduces the application of Tropos methodology in the early requirements engineering phase. Section 4 explains the late requirements generation process. Section 5 presents our proposed method of requirements elicitation and finally, Section 6 concludes the paper.

2. Overview of the Proposal
In this section, we present an overview of our proposed method to elicit and improve requirements list based on the organizational model. The complete method consists of several phases which allow us to elicit the requirements list provided by the clients from its organizational model.

The organizational model is the original input model, which is transferred to the SS-BM by the FELRE pattern language. The SS-BM is mapped to the requirements list for elicitation. The mapped SS-BM and requirements list are analyzed in specific sequence according to multiple checking points. The problems with the requirements list found during the analysis process will be solved based on the listed guidelines. At the end of the elicitation process, the quality indicators are used to assess the quality of requirements. This considerably improves the overall quality of requirements.

2.1 The Case Study

In order to illustrate our approach, we have used the Library Management System case study. Figure 2 shows the organizational model of this case study. Figure 1 shows the requirements list provided by the clients at the beginning of the project. In the following sections, we will demonstrate how to create the SS-BM from the organizational model (Figure 2) and how to use the SS-BM (Figure 3) to verify the requirements list and elicit further requirements. The motivation behind the entire process is to elicit requirements which are closer to the stakeholders’ perception.

The library manager has provided the requirements list (Figure 1) describing the required functionality of the software system to be developed. According to the requirements list, he/she wants to realize six main functions: Circulation management, Purchase management, Generate purchase plan, E-book management, Provide E-books and Students information maintenance.

According to the original organizational model (Figure 2), the library needs to realize the circulation management, purchase management and e-book management tasks. Circulation management includes borrowing and retuning books. Purchase management includes generating purchase plan, purchasing books and books processing. The library also provides the online e-book facility. Students’ information is stored in the Student Center. Library staff generates the purchase plan based on the book list provided by the bookshop. The concept of the organizational model will be explained in section 3.

3 Early Requirements

Each software system is developed to fulfill the stakeholders’ needs/requirements. In early requirements phase, analysis is based on understanding the organizational context in which the future system will be situated and capturing stakeholders intentions and rationale. The Tropos methodology has been effective in identifying and analyzing stakeholders’ intentions and rationale [5] [6] [7] [9] [10]. Stakeholders in a given domain are represented as actors(s). Stakeholders’ intentions are modeled as goals. Actors make use of plans and resources to realize their goals. The key concepts in Tropos for modeling early requirements include the following concepts [10]:

Actor: Actor is an entity, which presents a physical or a software agent as well as a role or a position. An actor carries out actions to achieve the goals.

Goal: Goal represents the strategic interest of an actor.

Plan: Plan represents the way of doing something at an abstract level.

AND-OR Decompositions: Decomposition represents the relationship between the root plans/plans and their sub-plans/sub-goals, which can be AND or OR.

Resource: Resource represents a material or an informational entity.

Dependency: Dependency represents the relationship between the two actors. One actor depends on the other in order to achieve some goal, execute some plan, or deliver a resource. The former actor is called depender, while the latter one is called dependee. The object between them is called dependum.

In the proposed case study, Tropos organizational models (Figure 2) are used to represent the early requirements. The three actors are Library, Student Center and the Bookshop. The main goal of the actor Library is Library Management, which has three sub-tasks: Manage Circulation, Manage E-Books and Manage Purchase. The task Manage Circulation has two AND Decomposed tasks: task Borrow Books and task Return Books. Actor Library depends on the actor Student Center to provide resource Student Information and actor Bookshop to provide resource Book List. Actor Library is the depender. Resource Student Information is the dependum. Actor Student Center is the dependee.

4. Late Requirements

As stated earlier, Tropos is a methodology, which is intended to support five phases of software development: early requirements, late requirements, architectural design, detailed design and implementation [10]. Late requirements models describe the functions and qualities of software system within its operational environment [3]. Organizational model is the output of early requirements, which represent the intentions of the stakeholders and organizational context in which future system will be situated. There is still a significant gap between the organizational model (early requirements) and requirements models (late requirements). To reduce the abstraction level between early requirements and late requirements, we employ the pattern language called “FELRE” [1] to transform the original organizational model to a new
Software System-Business Model (SS-BM), which integrates the Software System Actor (SSA). The new SS-BM is a late requirements model, which focuses on representing software functionality. The SS-BM will eventually be used in the requirements list elicitation.

4.1 Implementing the EFLRE pattern language
To implement the EFLRE pattern language, the following three steps were performed [1][2].
1. Identify the relevant plans to be automated. In our case study, except plan Purchase Books and Stick Book Barcode (shown as shaded plan), all the other plans need to be automated. Figure 2 shows the identification condition.
2. Place the SSA into the new organizational model. In this process, the actors that have some plans, goals or dependency relationship (to be automated) are included. In our case study, the actors Library, Student Center and Bookshop were included.
3. Transfer the plans or goals to be automated to the SSA. Table 1 shows a brief description of EFLRE pattern language, which consist of five patterns. By applying the transformation rules defined by “FELRE”, organizational model shown in Figure 2 was transformed to SS-BM shown in Figure 3. In our case study, the first four patterns were implemented as below:

a) The Final Plan without dependencies Automation Pattern
   In the case study, the plan Manage E-Books complied with the characteristics of the Final Plan without dependencies Automation Pattern (Figure 2). Figure 3 shows the results of the application of the pattern. The plan was transferred to the SS-BM and a new plan dependency – Input E-Books, between the actor Library and the SSA was generated.

b) The General Plan or General Automation Pattern
   In our case study, the goal Library Management complied with the characteristics of the General Plan or General Automation Pattern (Figure 2). The plans, such as Manage E-Books, Manage Circulation and Manage Purchase, are sub-plans of the goal Library Management. Figure 3 shows the results of the application of the pattern. Since these sub-plans had been transferred to the SSA, the goal Library Management was also transferred to the SSA.

c) The Depender-Dependee Actor Plans Automation Pattern
   In our case study, the plan Check Student Information complied with the characteristics of the Depender-Dependee Actor Plans Automation Pattern (Figure 2). The plan Check Student Information acts as the depender. The resource Student Information is the dependum. The plan Send Information (Student Center) is the dependee. Figure 3 shows the results of the application of the pattern. The plan decomposition is created. The plan

Check Student Information is the parent node. The plan Get Student Information is the child node, which depends on the actor Student Center to provide Student Information.

d) The Depender Actor Plan Automation Pattern
   In our case study, the plan Generate Purchase Plan complied with the characteristics of the Depender Actor Plan Automation Pattern (Figure 2). The Generate Purchase Plan is the depender, which was automated. Figure 3 shows the results of the application of the pattern. Actor Library still depends on the actor Bookshop to provide resource Book List. Plan Generate Purchase Plan depends on the actor Library to enter book information (task Enter BookInfo).

5 The Elicitation Process
5.1 Requirements lists
Requirements list (based on [11]) is provided by the customer to describe the business needs at the beginning of a project, which is described in natural language. Then analysts will communicate with the customers and users based on the requirements list to generate the requirements specification. In the end, the analysts will design software based on the requirements specification. So, the correctness and completeness of requirements list have significant impact on the software quality. We propose a method to evaluate and elicit requirements list based on the SS-BM. Figure 1 shows part of a Library Management System (LMS) requirements list.

<table>
<thead>
<tr>
<th>1. Circulation management</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1. Students can borrow books.</td>
</tr>
<tr>
<td>1.1.1. Check the inputted student information.</td>
</tr>
<tr>
<td>1.2. Students can return books.</td>
</tr>
<tr>
<td>2. Purchase management</td>
</tr>
<tr>
<td>2.1. New book processing</td>
</tr>
<tr>
<td>2.1.1. Input book information and scan book barcode</td>
</tr>
<tr>
<td>3. Generate purchase plan.</td>
</tr>
<tr>
<td>4. E-book management</td>
</tr>
<tr>
<td>5. Provide E-books</td>
</tr>
<tr>
<td>6. Students information maintenance</td>
</tr>
</tbody>
</table>

Figure 1. Library Management System Requirements List

A requirements list consists of hierarchical items, which are known as requirements items. For example, requirements item 1. Circulation management is the parent node of requirements item 1.1. Students can borrow books and requirement item 1.2. Students can return books.
5.2 Mapping requirements items with SS-BM elements

Analysts map each requirements item into SS-BM elements according to their semantic meanings. Figure 3 illustrates the semantic mappings from the requirements items to SS-BM elements. The instance, SS-BM used to specify semantic mappings from the requirements items to SS-BM elements is called \textit{map item}. For example, the requirements item \textit{1.1. Students can borrow books} was mapped to SS-BM element \textit{Borrow Books} by \#2 \textit{map item}. In the SS-BM, the element mapped to requirements items is called \textit{mapped SS-BM element}. The SS-BM element \textit{Borrow Books} is a mapped SS-BM element. In the requirements list, the requirements items mapped to SS-BM elements is called \textit{mapped requirements item}. Hence, the requirements item \textit{1.1. Students can borrow books} is a mapped requirements item.

In the mapping process, one requirements item might be mapped to many SS-BM elements by one map item. For example, the SS-BM elements \textit{Check Student Information} and \textit{Get Student Information} were mapped to requirements item \textit{1.1.1. Check the inputted student information} by \#3 \textit{map item}. More than one requirement items might be mapped into one SS-BM element as well. For example, requirements item \textit{4. E-book management} and \textit{5. Provide E-books} were mapped to SS-BM element \textit{Manage E-book} by \#9 \textit{map item}.

At the end of mapping process, all the semantic relationships between the requirements items and the SS-BM elements have been identified and indicated by the \textit{map items}.

5.3 Requirements elicitation and verification

The requirements will be improved by analyzing the mapping between requirements list items and SS-BM elements. The requirements list and SS-BM were analyzed separately according to different checking points. For each selected requirements list items, \textit{Correctness Checking} is conducted. The \textit{Correctness Checking} aims to check whether the requirements item was mapped to an SS-BM element. For every SS-BM elements, the \textit{Completeness Checking}, \textit{Parent/Child Hierarchical Consistency Checking} and \textit{Peer/Peer Hierarchical Consistency Checking} has to be conducted. The \textit{Completeness Checking} aims to check whether the SS-BM element was mapped to requirements items or not. The \textit{Parent/Child Hierarchical Consistency Checking} aims to check whether the parent/child hierarchical relationship of the SS-BM element and its mapped requirements item is consistent. The \textit{Peer/Peer Hierarchical Consistency Checking} aims to check whether the peer/peer hierarchical relationship of the SS-BM element and its mapped requirements item is consistent.

In this section, each one of the four checking points are presented with examples, the problems found by the checking process and the solutions of the problems are also explained.

- **The requirements list was analyzed from beginning to the end. Each requirement item was checked in this step.**

  **Correctness Checking:** Check whether the requirements item was mapped to any SS-BM elements.

  **Problem:** The requirements item might not be mapped.

  **Solution:** Remove the requirements item from requirements list.

  **Example:** In our case study, the requirements item \textit{6. Students information maintenance} was not mapped to any SS-BM element. This requirement item is not necessary and will be removed. Actually, according to the SS-BM, the student information doesn’t need to be maintained by the Library System. It is to be supplied by the actor \textit{Student Center}.

- **The SS-BM was analyzed from top to down, left to right. Each SS-BM element was checked in this step.**

  **Completeness Checking:** Check whether each SS-BM element was mapped into any requirements items.

  **Problem 1:** The SS-BM elements might not be mapped.

  **Solution 1:** Add a new requirements item into the requirement list to present the SS-BM element. The new requirements item needs to be added under the requirements item, which is mapped to the SS-BM element’s parent node.

  **Example 1:** In our case study, the SS-BM element \textit{Update Book Status} was not mapped to any requirements item. A new requirements item \textit{1.1.2 Update book status} will be added below the requirements item \textit{1.1. Students can borrow books}, which is mapped to the SS-BM element \textit{Borrow Books} - the parent node of the SS-BM element \textit{Update Book Status}.

  **Problem 2:** The SS-BM elements might be mapped to more than one requirement items.

  **Solution 2:** Combine the two requirement items mapped to one SS-BM element into one requirement item.

  **Example 2:** In our case study, the SS-BM element \textit{Manage E-books} was mapped to requirements item \textit{4. E-book management} and \textit{5. Provide E-books} by \#9 \textit{map item}. These two requirements items were
combined into one requirements item 4. E-book management.

**Parent/Child Hierarchical Consistency Checking:** Check whether the requirements item, which was mapped to the current checking SS-BM element’s parent node, is the parent node of the requirements item that the current checking SS-BM element is mapped to.

**Problem 1:** The current checking SS-BM element and its parent node were mapped to the same requirements list item.

**Solution 1:** The requirements item need to be split into two requirements items. One of the new requirements items is the other one’s parent node.

**Example 1:** In our case study, the SS-BM element Check Student Information and Get Student Information were mapped to requirements list 1.1.1. Check the inputted student information. This requirements list will be split into 1.1.1. Check whether it’s a legal student and 1.1.1.1. Get student information from Student Center.

**Problem 2:** The requirements item, for which the current checking SS-BM element’s parent node was mapped to, is not the parent node of the requirements item that the current checking SS-BM element is mapped to.

**Solution 2:** The requirements item, which the current checking SS-BM element mapped to will be moved under the requirements item, which was mapped to the current checking SS-BM element’s parent node.

**Example 2:** In our case study, the plan Manage Purchase was mapped to requirement item 2. Purchase management by #5 map item. The SS-BM element Book Processing is mapped to requirements item 3. Generate purchase plan by #6 map item. The SS-BM element Manage Purchase is the parent node of SS-BM element Book Processing. In the requirements list, the requirements item 3. Generate purchase plan will be moved under requirements item 2. Purchase management as 2.2. Generate purchase plan.

**Peer/Peer Hierarchical Consistency Checking:** Check whether the current checking SS-BM element and the SS-BM elements in the same hierarchical level were mapped to the requirements items in the same level.

**Problem:** The current checking SS-BM element and the SS-BM elements in the same hierarchical level might be mapped into requirements items. Another problem is the current checking SS-BM element and the SS-BM elements in the same hierarchical level might be mapped into requirements items that has different parent node. This hierarchical problem is handled by Parent/Child Hierarchical Consistency Checking.

**Solution:** The requirements item will be split into two requirements items to represent the two SS-BM elements.

**Example:** In our case study, the SS-BM elements Input Book Info and Scan Book Barcode were mapped to requirements list 2.1.1 Input book information and scan book barcode by #7 map item. This requirements list will be split into requirements list 2.1.1. Input new book information and requirements list 2.1.2 Scan new book barcode.

Figure 1 (A) shows the new library management system requirements list, after performing the above mentioned verification and elicitation processes.

<table>
<thead>
<tr>
<th>1.</th>
<th>Circulation management</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1.</td>
<td>Students can borrow books.</td>
</tr>
<tr>
<td>1.1.1.</td>
<td>Check whether it’s a legal student</td>
</tr>
<tr>
<td>1.1.1.1.</td>
<td>Get student information from Student Center</td>
</tr>
<tr>
<td>1.1.2.</td>
<td>Update book status</td>
</tr>
<tr>
<td>1.2.</td>
<td>Students can return books.</td>
</tr>
<tr>
<td>2.</td>
<td>Purchase management</td>
</tr>
<tr>
<td>2.1.</td>
<td>New book processing</td>
</tr>
<tr>
<td>2.1.1.</td>
<td>Input book information</td>
</tr>
<tr>
<td>2.1.2.</td>
<td>Scan book barcode</td>
</tr>
<tr>
<td>2.2.</td>
<td>Generate purchase plan.</td>
</tr>
<tr>
<td>3.</td>
<td>E-book management</td>
</tr>
</tbody>
</table>

Figure 1 (A). New Library Management System Requirements List

**5.4 Quality Indicator**

After the elicitation process, the problems in the requirements list can be discovered. We use three measurable *quality indicators* to summarize the quality of the requirements list in numerical order based on [11]. The goal of [11] is different to ours though, where domain ontology can be used as domain knowledge. The three quality indicators are Correctness indicator, Completeness indicator and Hierarchical Consistency indicator. The Correctness indicator and the Completeness indicator reflect the Correctness Checking result and the Completeness Checking result respectively. The Hierarchical Consistency indicator reflects the result of Parent/Child
Hierarchical Consistency Checking and Peer/Peer Hierarchical Consistency Checking.

**Correctness (CO):** The proportion of requirement items that were mapped into SS-BM elements represents the correctness of the requirements list. The Correctness Checking result can be reflected by this indicator.

\[
CO = \frac{\# \text{ {requirements items that are mapped into the SS-BM elements}}}{\# \text{ {requirements items}}}
\]

In our case study, there are total of eleven requirements items. In these eleven requirements items, nine of them were mapped to the SS-BM elements. For example, the requirements item 1. Circulation management was mapped to SS-BM element Manage Circulation by #1 map item. There is only one requirements item - 6. Students information maintenance failed in the Correctness Checking, which was not mapped to any SS-BM element. Hence, CO = 10/11 = 91%

**Completeness (CP):** The proportion of SS-BM elements that were mapped to the requirements items represent the completeness of the requirements list. The Completeness Checking result can be reflected by this indicator.

\[
CP = \frac{\# \text{ {SS-BM elements that are mapped onto the requirements items}}}{\# \text{ {SS-BM elements}}}
\]

In our case study, in the total there were twelve SS-BM elements. One SS-BM element - Update Book Status was not mapped to any requirements items. Another SS-BM element - Manage E-Books was mapped to two requirements items. These two SS-BM elements failed the Completeness Checking. Hence, CP = 10/12 = 83%

**Hierarchical Consistency (HCST):** Two types of hierarchical relationships were used to describe the relationships between two SS-BM elements or between two requirements items. They are parent – child relationship and peer – peer relationship (as described previously in the paper). The proportion of hierarchical relationships in SS-BM that are not conflicting with the hierarchical relationships in requirements list represent the hierarchical consistency of the requirements list. The Parent/Child Hierarchical Consistency Checking and Peer/Peer Hierarchical Consistency Checking result can be reflected by this indicator.

\[
HCST = \frac{\# \text{ {hierarchical relationships in SS-BM not conflicting with the hierarchical relationships in requirements list}}}{\# \text{ {hierarchical relationships in SS-BM}}}
\]

In our case study, there are thirteen hierarchical relationships in SS-BM. For example, the relationship between SS-BM element Borrow Books and element Check Student Information is parent – child. Elements Borrow Books and Update Book Status also have parent – child relationship. The relationship between element Check Student Information and element Update Book Status are the peer-peer relationship. There are totally three relationships between these three SS-BM elements. The SS-BM elements Borrow Books and Check Student Information were mapped to requirements item 1.1. Student can borrow books and 1.1.1. Check the inputted student information respectively. The SS-BM element Update Book Status was not mapped to any requirements item. In other words, there is only one relationship between the requirements items that the three SS-BM elements were mapped to. The relationship between requirements items 1.1. Student can borrow books and 1.1.1. Check the inputted student information is a parent – child relationship, which complied with the relationship of the related SS-BM elements in the SS-BM. In these three SS-BM elements, the HCST equals 33% (1/3).

In the whole case study: HCST = 7/13 = 54%

**6 Conclusion**

In this paper, a method to elicit requirements from SS-BM is proposed. The SS-BM is derived from organizational model by implementing FELRE pattern language. The proposed method consists of three main processes. In the first step, the SS-BM elements and requirements items are connected by map items in a semantic way. In the second step, the mapped requirements list and SS-BM were analyzed in specified sequence. Each requirements item and SS-BM element was analyzed according to multiple checking points. In the end, the quality indicators are used to estimate the quality of the requirements elicited. In the future, we are working towards automating the proposed approach. We shall be conducting experiments to evaluate the proposed method.

**References**

1. Martínez, A., Pastor, O., Estrada, H. A pattern language to join early and late requirements. In Journal of Computer Science and Technology, special issue on


<table>
<thead>
<tr>
<th>Pattern Name</th>
<th>When to apply the pattern</th>
<th>Plan or goal to be automated</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Final Plan without dependencies Automation Pattern</td>
<td>When a final plan without dependencies needs to be automated, apply this pattern.</td>
<td></td>
</tr>
<tr>
<td>The General Plan or General Automation Pattern</td>
<td>When a General Plan or General Goal needs to be automated, apply this pattern.</td>
<td></td>
</tr>
<tr>
<td>The Depender - Dependee Actor Plans Automation Pattern</td>
<td>When the plans to be automated are both the depender actor plan and the dependee actor plan, apply this pattern.</td>
<td></td>
</tr>
<tr>
<td>The Depender Actor Plan Automation Pattern</td>
<td>When the depender actor plan must be automated, apply this pattern.</td>
<td></td>
</tr>
<tr>
<td>The Dependee Actor Plan Automation Pattern</td>
<td>When the dependee actor plan must be automated, apply this pattern.</td>
<td></td>
</tr>
</tbody>
</table>

Table 1 A short description of the FELRE patterns [1]

Figure 2: Library Management Organizational Model
Figure 3: Library SS-BM mapped with Requirements List