A Client Oriented Requirements Baseline

Julio Cesar Sampaio do Prado Leite
Departamento de Informática, PUC-Rio
R. Marquês de S. Vicente 225
Rio de Janeiro 22453-900 Brasil
julio@inf.puc-rio.br

Antonio de Pádua Albuquerque Oliveira
Serinf, Petrobrás
Av. Chile 65 sl. 1556
Rio de Janeiro 20035-900 Brasil

Abstract

Traceability, a major issue in software engineering, is seldom present at the initial requirements engineering process. This paper reports on a proposal for organizing requirements statements as a model, where change and evolution are taken into consideration. The model uses natural language statements as its basic representation, which helps the communication between clients and software engineers. The model is supported by a customized software system for which a prototype was built and used in an industrial setting.

1 Introduction

A recent survey on traceability [4] has pointed out that most of the research and use of traceability methods and tools does happen after the availability of a software specification. It was also observed that the existing traceability practices fail to trace the origin of the requirement. It is not necessary to argue about the importance of traceability, but it seems that by not identifying and working at the starting point, the results of traceability are less productive than they could be.

It is our opinion that the lack of traceability towards the real origins of the software is mainly due to the lack of study in the requirements process area. Requirements engineering has been paying too much attention on the modeling of products, and only recently has been looking into the aspect of requirements generation and its intrinsic dependency on the social aspects that surrounds and will surround the software.

Our proposal is centered on a client oriented baseline. This baseline is a structure which incorporates sentences about the desired system. These sentences are written in natural language following defined patterns. The sentences are written by software engineers but are validated by clients. The supporting software system, a prototype, is a specialized editor with hypertext navigation capabilities and a versioning system.

The basic idea behind the baseline is that it is perennial. It is developed during the requirements engineering process, but keeps evolving as the software construction process evolves. Figure 1 gives an idea of the parallelism between the evolution of the requirements baseline and the evolution of the software product. Our baseline is independent of the process model used in the development, but in Figure 1 we illustrated it using the classical waterfall model.

In this paper we focus on the conceptual model used as basis for the baseline specification and give some initial impressions of its use, based on the experience of a trip authorization system of a large oil company. The process aspect of building and using the baseline is dealt in another report [10].

2 Assumptions

Our baseline is geared towards "external requirements" that is requirements as posed by the clients of the macrosystem to which software will be one of the parts. External requirements does not consider "operational constraints", that is constraints that will
emerge later on at the software construction process. These operational constraints may be derived from factors like technology availability, cost decision, and other software design decisions. It is important to note that this will impact the baseline, since requirements can be classified into included (considered) or excluded (not considered) [10]. The external requirements could be compared to what McMenamin and Palmer [9] call "true requirements", that is they do not take in consideration the supporting technology (hardware and software).

It is obvious that the requirements process is a cooperation effort involving users, clients, software engineers, consultants, experts, and so on. These actors work in different social contexts, have different backgrounds and different viewpoints. We acknowledge this fact and assume that there is a previous elicitation process that will enable the software engineers to write down requirements statements.

We believe that the process of defining something, always depend on a previous design. So, the external requirements to the software are a result of an overall design proposed by information system engineers or by systems engineering engineers. We also should not forget that as the external requirements are proposed and analyzed, they may impact the previous design. The information system engineers and/or the systems engineering engineers work in and at the same time, help to establish what we call the Universe of Discourse\(^3\). Figure 2 shows a SADT datagram [12] which portrays our basic context assumptions.

Our philosophy is that "all the requirements have to be recorded" and we wish to help answer the question: "Which facts have to be elicited from the Universe of Discourse and validated by clients to serve as basis for a software specification?"

The clients oriented baseline is built upon previous work [1], [9], [5] [8]. Carvalho [1] claims that identifying the clients' actions is the best anchor for guaranteeing the effectiveness of an information system. Clients' actions are the day to day activities which can be expressed by an infinitive verb. Carvalho's idea is very similar to the Jackson's actions [5]. McMenamin and Palmer [9] made a strong argument of the effectiveness of an outside-in strategy as compared to a top-down one when trying to identify true requirements. The backbone of this strategy are the external events. External events are organized as lists [8] and should model the set of all the needs which constitute the origins of the problem to be solved by the system and also map all the perceived interactions between the system and the Universe of Discourse.

Our work so far has used information systems examples, and as such we can not argue about the use of our model in system engineering requirements.

\(^3\)The Universe of Discourse is the overall context in which software will be developed and operated. The Universe of Discourse includes all the sources of information and all the people related to the software. These people are referred as actors in this UoD. It is the reality trimmed by the set of objectives established by those demanding a software solution.

![Figure 2: Context Assumptions](image)

3 The Baseline Conceptual Model
Our baseline has three main views: the configuration view, the basic model view and the hypertext view.

3.1 Basic Model View
The basic model uses the entity relationship framework [3] as a representation language. Figure 3 shows the entity relationship diagram of the basic model behind the baseline. Below, we describe the entities pertaining to the ER model.

- **Client** - Client is a person or group who is responsible for **Action**. It has the following structure:
  ```
  Name + Location
  EX: Financial Department at main office. <CLO1> (Name) (Location)
  ```

- **Action** - Is the concrete action taken by **Client** in their work. An action is represented by a sentence with the following structure:
  ```
  Infinitive Verb + Predicate
  + [Father-Action-ident] + [Constraint-attribute] + [Diagnosis-attribute]
  ```

  An action can be decomposed in sub-actions with the same structure of an action.
  EX: `<CLO1> Pay perdiem stipend of a NIA, (Inf. Verb) (Predicate) safely and timely (constraints) since perdiem stipend is not automatic
- **Temporal Stimuli** - Instants of time where the system will have to react independently of an Input. It has the following structure:

  Time reference + Subject + Verb + Predicate

  Ex: At the end of the month, soliciting sector (time reference) (subject) needs NTA's report. \(<TSD1>\) (verb) (predicate)

- **External Stimuli** - A happening in the Universe of Discourse that will require a reaction from the system. It has the following structure:

  Subject + Verb + Predicate

  Ex: Soliciting sector reports (subject) (verb) a business trip. \(<ESO1>\) (predicate)

- **Output** - An information structure that will be produced by the system. It has the following structure:

  Name + (Description) + {Constraint-attribute} + (Diagnosis-attribute)

  Ex: \(<TSD1>\) Pending NTA (Name) must have pending information, (constraint) since it is hard to find out the problem. (diagnosis)

- **Input** - An information structure that is consumed by the system. It has the following structure:

  Name + (Description) + {Constraint-attribute} + (Diagnostic-attribute)

  Ex: \(<ESO1>\) NTA (Name) must be filled by soliciting sector, (constraint) since it is difficult to use financial codes. (diagnosis)

The most important attributes of our ER model are the Constraint and the Diagnosis attributes. A Constraint is a scope or quality requirement referring to a given entity. It is represented by a short sentence with the following structure: MUST + Verb-c + Predicate. A Diagnosis is an observation that raises a problem related to a given entity. It is represented by a short sentence, or by a small paragraph. Diagnoses are the central point of client validation [10].

The terms: Name, Location, Subject, Verb and Predicate can be chosen from the Language Extended Lexicon [7] table, a structure that makes it possible the
usage of a controlled vocabulary. Observe that Name may have the same instance as Subject or Verb, for example a term Name in Client can be the Subject of an External Stimuli.

3.2 The Configuration View

Figure 4 shows another view of our baseline. This view is the versioning system behind our basic model (Figure 3). Our versioning keeps the history of changes to the base, keeping the following main information: date, time, person making the change (user), reasons for the change (change trigger, date of trigger, authorization) and type of change (input, modification, exclusion).

The consistency of the configuration is warranted by consistency constraints determined by the basic model. As such, change operations trigger a process for consistency checking that is responsible for the consistency of a given configuration. If, for instance, an external event is excluded and its dependent entities are not, the configuration scheme must provide a warning regarding the lack of consistency in the baseline or not allow the operation without exclusion of the dependent entities.

3.3 The Hypertext View

Figure 5 shows the hypertext view of our baseline. The hypertext links are determined by the relationships in the basic model and by the usage of a controlled vocabulary. The controlled vocabulary is dependent on the user, who in editing time can choose the terms: Name, Location, Subject, Verb, Predicate from an existing table of Language Extended Lexicon [7] entries (LEL-table). This table is available before the writing of requirements. If a term is not present in the table, and the user believes it should be, the term is entered on a delta table (Delta-table) which will contain candidates for further inclusion in the Lexicon. The LEL-table and the Delta-table keep a link to where the word is used, so enabling the navigation through common terms.

4 The Example

The example is based on a real information system, a business trip system, developed at Petrobras (The Brazilian Oil Company). The information system objectives are: to automate procedural aspects of the system and provide information about business trips in the country. Since the original baseline was written in Portuguese, and we performed a free context translation, the terms presented in the example figures may be a little awkward.

We exemplify the business trip system baseline providing a partial instantiation of the three views presented above. Figure 6 shows an ER diagram in which the action "Solicit a business trip" is partially detailed. Figure 7 uses an instance of the actions entity to exemplify the versioning mechanism of the configuration view. Figure 8 shows the links created by vocabulary control.

As said before the organization of the baseline requires a prior elicitation process [10] in which the Language Extended Lexicon [7] is built. We show below three entries of the business trip system LEL.

NTA/ National Trip Authorization

• Notion:
  – A trip request document filled in by a soliciting sector.

• Behavioral Response:
  – It must be signed by an authorized supervisor.

Solicit a Business Trip

• Notion:
  – An action taken by a soliciting sector every time there is a business trip.

• Behavioral Response:
**Figure 5: The Hypertex View**

**Figure 6: An Instantiated ER model for the baseline**

**Figure 7: An Instantiated Versioning View**

<table>
<thead>
<tr>
<th>TERM</th>
<th>ENTITIES - ADDRESSES</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;xyz&quot;</td>
<td>client-01</td>
</tr>
<tr>
<td></td>
<td>event-05</td>
</tr>
<tr>
<td>&quot;tim&quot;</td>
<td>client-02</td>
</tr>
<tr>
<td></td>
<td>event-10</td>
</tr>
<tr>
<td>&quot;jfs&quot;</td>
<td>external event-15</td>
</tr>
<tr>
<td>&quot;vvlm&quot;</td>
<td>client-01</td>
</tr>
<tr>
<td></td>
<td>event-05</td>
</tr>
<tr>
<td>&quot;it&quot;</td>
<td>input-15</td>
</tr>
<tr>
<td></td>
<td>action-02</td>
</tr>
</tbody>
</table>

...
- An NTA must be filled.
- An NTA must be signed by an authorized supervisor.

**Soliciting Sector:**

- **Notion:**
  - A company sector which solicits a business trip.

- **Behavioral Response:**
  - The soliciting sector code must be the one defined by the org system.
  - It can be a sector different from the employee's sector.

Figure 8 exemplifies a possible LEL table, where the terms are instantiated as follows: xyz is a client-name, tlm is a client-location, jsl is an action-verb, vvlm is an external stimulus-subject and it is an action-predicate. Observe in Figure 8, that the use of vocabulary control creates links not determined by the basic ER model. The use of this vocabulary scheme makes it possible that instances not related of related entities be linked by common vocabulary (external event-15 and action-10 are not related by the "have" relationship, but since they use a common verb, a link is available between the two instances).

5 **Usage**

Petrobrás has been using, for some time, the idea of clients' actions [1] in the definition of information systems. This concept, centered on real actions taken by managers, has helped the design of better systems at the organization. Our proposal is centered on the clients' actions idea.

Our research goal was to find a baseline structure, based on the concept of clients' actions, that could be used by information engineers, software engineers and clients. The ER model was used as the infrastructure for our work and it has passed through several revisions, before we decided on the current one (Figure 3).

Model validation was performed by trial and error, using Petrobrás business trip system, which used the clients' actions concept. Using the ER model and the idea of the LEL, we specified the hypertext view. The configuration view was specified based on fundamentals of configuration management.

Our approach towards the validation of the baseline and technology transfer has been the following. First we have written, without automation, a complete ER view for the business trip system. Second we have developed the first prototype, mainly with editing capabilities, and a weak versioning support. Instead of hypertext navigation we used fixed queries based on the basic relationships identified by the ER model. This first prototype was shown to systems analysts and users whom used it and provided a superficial feedback. We are currently finishing a new version of the tool.

We agree that before using a tool we need to have understood the method for using the tool. In our case: we have the method, a first prototype and are working on a next prototype version, but to successfully use our baseline the tool is essential. Tracing requirements by hand, as we did in the ER validation, is a arduous task and counter productive. Without automation, traceability is a myth.

The first prototype, here described by a JSD [5] implementation diagram (Figure 9), was built around an editor for each entity. The editor has the basic capabilities of inclusion, exclusion, and modification for each entity. Each entity requires user identification, and may require constraints and diagnoses. At editing time the user would link instances of related entities. The system provides five types of reports which are to be read and validated by clients.

6 **Related Work**

Our baseline is related to previous work [6] [7] mainly due to the use of the Language Extended Lexicon. In [6] we have proposed the use of requirements sentences together with LEL to represent internal requirements, where the requirements sentences were classified in three classes (input, state change, and output) and with the terms defined in the LEL. The baseline is a more complex structure and takes in consideration, besides vocabulary control, the aspects of navigation and configuration management.

The baseline, besides providing the possibility of forward traceability, stresses the pre-requirements traceability [4] by means of diagnoses and keeping the reasons for change in the configuration view. Diagnoses serve as a work area for posing and solving problems, and are the basis for the validation cycle of our baseline [10].

Potts [11] has been exploring pre-requirements traceability using the IBIS framework [2] to keep the history of requirements formation and the negotiation process related to it. Our proposal can also be classified as pre-requirements following Gotel’s classification. Although paying more attention on a systematic representation and less on idea formation, it is still far from a system specification. It is geared towards external requirements and supports a validation cycle between requirements writers (software engineers) and requirements readers (clients). The concept of diagnoses is also important in providing a link to real world aspects.

7 **Conclusion**

We reported on a conceptual model for a client oriented requirements baseline. The usage of natural language expressions makes it accessible to clients users. The conceptual model uses three different viewpoints: an entity relationship view, a configuration view and a hypertext view. The entity relationship view is centered on the idea of clients' actions, a strategy that has been used successfully at Petrobrás. The hypertext view is based on the idea of controlled vocabulary and the configuration is based on the configuration literature and on the IBIS work.
Figure 8: The Instantiated Hypertext View

Figure 9: First Prototype Implementation Diagram
The combination of three different representation schemes together with a natural language based interface is, in our opinion, a strong support in order to enact a traceability process. Nonetheless, we have to understand that this process is a social process; different actors with different roles will have to cooperate in order to trace the needed information. This cooperation depends not only on technologies, which our baseline is an instance of, but also on policies and procedures used by the organization.

Work at Petrobrás is continuing. It is hard to introduce new technologies in large companies. A new version of the prototype is being built and a new case study is being planned. Although technology transfer is a hard task, we believe we are trailing the usual steps towards technology acceptance.

In terms of research we are planning to explore the forward traceability from our baseline to established specification methods. We should be working on integrating the baseline with Maffeo’s specification method [8], which is an evolution of McMenamin and Palmer’s Essential Analysis [9], and with the OMT method [13].

Acknowledgement

We wish to thank Bruno Maffeo for his comments and suggestions on an early version of this paper.

References


