Requirements Management in Large Software System Development

Sooyong Park[1] and Jongho Nang
Department of Computer Science
Sogang University, Seoul, Korea
{sypark, jhnang}@ccs.sogang.ac.kr

Abstract

The management of requirements is an essential element of software development to ensure program success. As software systems become increasingly large, the management of their requirements becomes increasingly challenging. This paper summarizes an approach to software systems requirements management, called "engineering baselines" that has been developed to address those challenges. There are three primary dimensions of those requirements management challenges as addressed by the engineering baseline approach:

- *Machine processing of system text, especially the requirements statements,*
- *Correlation [tracing] between requirements and all other system elements, including hardware components, code modules, test cases, user manuals, and design specifications,*
- *Change consideration and control.*

There are three principal features of the engineering baseline approach:

- *Structuring [and decomposing to lowest level] requirements statements [and all other system lists, descriptions, designs, etc.] into separate, autonomous, stand-alone statements called system elements,*
- *Numbering, with a unique system number to each system element,*

Maintaining history of changes and correlating [tracing] all system elements using their unique system numbers

1. Introduction

As computer based system has been involved larger application domain, management of these system development has been more complicated and sometimes uncontrollable. The successful management of a large system development requires strict control over the requirements specification, and the documentation and code constituting the product [McD91]. In that management, requirements management becomes an essential element of software development to ensure program success. As software systems become increasingly large, the management of their requirements becomes increasingly challenging.

There are three aspects of current practice in requirements management for software system development that need to be addressed as those systems become increasingly large:

1. Limiting formal control of requirements to hard copy, formatted specifications, without associated "controlled" machine processable files in ASCII format, and only published/distributed at infrequent intervals after all "approvals" in place.
2. Use of both compound statements as well as bulleted and tabular data that are thus neither lowest-level statements nor autonomous standalone system elements--for separate allocation, tracing [correlation], testing, installation, etc.
3. Change management: change consideration in a controlled framework; change history correlation/tracing; and change impact assessment without a standard set of system numbers for all system elements and two-column ASCII formatted index files for change correlation [tracing].

This paper summarizes an approach to software systems requirements management, called "engineering baselines" that has been developed to address those challenges. The background and problems that we observed in large software system development are discussed in section 2. In section 3, the engineering baseline process will be discussed. The techniques used in engineering baseline approach will be discussed in section 4. An application of the approach and conclusion will be discussed in section 5 and 6 respectively.

2. Background and Problems

Separate groups are involved in large software systems; these include groups for modeling, test, design, and segment developments. The baselines available to these separate groups are generally limited to those that are formally approved and published, in hard copy--with change pages-- by a Configuration Management Office (CMO). One of problems in the formats used in such formal CMO requirements documents is that requirements numbering scheme is based on structured section number that is too broad to identify each requirement that is allocatable, testable, and traceable. An example "trace

[1] This research was supported by the Sogang university Research Grants in 1998.

table" from one large software system is presented in Table 1.

Table 1 Example Requirements Trace Table

System Concept Doc.	System Spec.	Function	Segment Spec A	Segment Spec. B
3.1.6	3.1.3	Support Transaction Control	3.5.1.1 3.5.1.2 3.7.1.1.1 3.7.1.1.2	3.2.8
3.1.7.1	3.1.4.1 3.1.4.2 3.1.4.3	Provide Training System Environment	3.2.1.2 3.2.1.5 3.2.8.5 3.4.2.1	3.2.3 3.4
3.1.7.2	3.1.3.3.7	Provide on-line help	3.2.2.5 3.2.6.2.2	3.4.5

There are two primary problems with this type of requirement structuring and tracing:

- level of trace -i.e. section level is too broad.
- ambiguous that higher level trace section means lower level section trace.

The "granularity level" problem makes it difficult to identify specific traces since sometimes more than a hundred requirements statements belong to a section. If it says those two section trace, that does not mean there are 10,00 (100*100) pairs of trace. If a developer interested in particular feature belong to a requirement statement, it is hard to traced requirements statement.

Another problem in using hard copy requirements specification is that they do not represent changes. The changes in requirements are more popular in large software system development. Some reason can be [Dav93]:

1. users really do not understand what to say and develop
2. developers really do not understand what they said and to describe

Therefore, while the development is in progress, some requirements become clearer, defective requirements are identified, and new or additional requirements are issued; these are typically called a Request For Change (RFC). RFCs go to configuration management office and review them with an impact and cost analysis to decide the acceptance. Due to the continuous RFCs, CMO can not issue updated requirements specification dynamically. They collect certain number of RFCs and reproduce new version of requirements specification. With new version of requirements specification, engineers can not get any information of what and how has been changed, history of changes.

We identified that follow supports are needed to resolve these problems;

1. organization of requirements statement to be represented as a unit,
2. assigning unique number to each organized requirements,
3. record keeping of all change history.

The engineering baseline approach enables the requirements specifications that contain unique identification number, history change, and allocation and trace information. The activities that produce and manage engineering baselines are called engineering baseline process.

3. Engineering Baseline Process

The activities that produces and manages engineering baselines are depicted in figure 1. The engineering baseline process includes:

1. organization,
2. numberization,
3. specification generation,
4. change management.

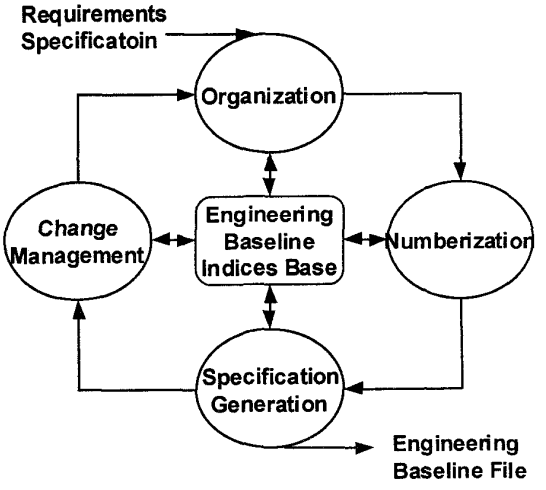

Figure 1. Engineering Baseline Process

Organization

This process establish standard structured ASCII representations of documentation - not bulleted and otherwise formatted that includes,

1. converting tabular data to whole statement
2. concatenating lead in information to bullets to make them independent standalone entities called system element
3. generating index of line sequence number and system element text

Numberization

Three types of numbers are used in this process. The most important is the system number, a unique identifier for every system element. Theses numbers establish machine-auditable allocations and links between elements, and record changes to them individually as well as their associations.

The engineering tag permits the definition of links among system elements, including the allocation of system elements to many types of categories, including elements of the Work Breakdown Structure (WBS). The change set numbers establish for each system element links to both the global version from which the element comes, and to each RFC that affects this element.

This process produces three index files that include index of line number and system number, system number and engineering tag, system number and change set number.

Specification Generation

System numbers, engineering tags, and change numbers are maintained in plain ASCII files at Engineering Baseline Indices Base. The specifications are generated by using these number indices to show various engineers options such as requirements statement with change history and requirements statement with trace tag.

Change Management

This process is to maintain system number history, associated RFC number information, and version number history. When RFCs are processed, new system numbers are issued to changed or new requirements statements. The newly issues system numbers are recorded into system number history file. The RFC number is recorded with associated system number. The new version number is recorded with these updates.

3.1 Organization

The main purpose of this process is to organize requirements statement as into separate, autonomous, stand-alone statements and issue line sequence number to each requirements statement. The typical example of non-stand-alone requirements is itemized requirements statements. As you can see in figure 2, each itemized requirements is meaningless without lead-in sentence.

3.1. The allocation of equirements to different functional areas is not meant to imply design; any solution that provides the required functionality is acceptable. The System will provide the following major functional capabilities:
a. Subject Search,
b. Ad hoc Subject Search,
c. File Maintenance,
d. Response Generation, and
e. Storage and Retrieval.

Figure 2. Example Requirements Statements

The example requirements statements of Figure 2 are results of organization process that produce two column index file of line sequence number and text. It is described in Figure 3.

110. 3.1. The allocation of requirements to different functional areas is not meant to imply design; any solution that provides the required functionality is acceptable.
111. 3.1. The System will provide the following major functional capabilities: a. Subject Search,.
112. 3.1. The System will provide the following major functional capabilities: b. Ad hoc Subject Search,.
113. 3.1. The System will provide the following major functional capabilities: c. File Maintenance,.
114. 3.1. The System will provide the following major functional capabilities: d. Response Generation, and.
115. 3.1. The System will provide the following major functional capabilities: e. Photo Storage and Retrieval.

Figure 3. Example of Line Sequence Number and Text (LT) Index

In this process, line sequence numbers are issued to each organized requirements statements called system elements. The line sequence number indicate the order of each system elements in the particular file. This number is not unique number that does not changed with system elements. The unique number, system number, is issued at numberization process.

3.2 Numberization

Three numbers are used in the engineering baseline. The first, and most fundamental, is a number for every system element: whether text, software code, drawing, hardware component, or model element. This is called the system number, and is cited as <<sn>> as a symbol of its use as <<xxxxxx.yyy>>. The second is an associated number, the engineering tag, [xxxxxx.y.zzzz]. The tag enables a numbered link of all system elements, citing the associated system numbers, and also a numbered allocation of all system elements. A third number set, the change numbers, <cxxxx> and <Cxxxx> embeds in each system element the citation of both the global change (version), designated by a sequential change number <cxxxx>, and the <Cxxxx> as the number of each individual Request for Change (RFC), Specification Change Notice (SCN) or other formal Configuration Control Board (CCB) type action.

From the discussed concept, each numbers can be represented as follows.

1. For system number <<n.d>>, n = n + 1 when system element is changed or new system element is added
   n = 0 as initial value
   d = d + 1 when new domain of system element is created.
   d = 0 as initial value

In our approach n is a six digit number and d is a three digit numbrt.

From this process, three index files are generated that include

2. For engineering tag [l.k.s], if l = n and n <<n.d>> it represent a link

k: System Element Type index
This character establishes type the system specification element. For example "c" indicate contractor need to cognize, "n" indicate does not need to be cognized in terms of system development.

s: Engineering tag distribution index
This "suffix" establishes the category of the xxxxxx tags. For example it may designate the links to the parent document, or it may establish that the "tag" is the allocation to test cases. The program office allocates blocks of tag suffixes to various organizations. For example the following block assignments are illustrative:

For example,
0000-0099 SEU (Systems Engineering Unit)
0100-0199 QAU (Quality Assurance Unit) etc.--

3. For change number <c>, c = RFC set serial number and RFC number <C>, C=RFC number,
<c> , c = c + 1 when system element is changed or new element is added c = 0 as an initial value
<C>, C = RFC number that indicate CCB approval

1. Line sequence number and system number (LS) index, SC
2. System number and engineering tag (SE) index, and
3. System number and change number (SC) index.

The example LS and SE index file of figure 3 is shown in figure 4[2]. As it indicates, they are two column ASCII files and stored into Engineering Baseline Indices Base.

Line Number	System Number	System Number	Eng. Tag
110	00110	00110	000000.n.0001
111	00111	00111	000000.c.0001
112	00112	00112	000000.c.0001
113	00113	00113	000000.c.0001
114	00114	00114	000000.c.0001
115	00115	00115	000000.c.0001

Figure 4. Example Index Files

3.3 Specification Generation

The specifications are generated by using index files that are stored at Engineering Baseline Indices Base to show various engineers options such as requirements statement with change history and requirements statement with trace tag. The index files that include LT, LS, SE, SC are combined to generate specification. The figure 5 shows the generated specification of the example used in figure 4. Based on the LT index, it attaches the system number by LS index. The change number and engineering tag are attached based on the SE and SC index files.

110. <c0000> 3.1. The allocation of requirements to different functional areas is not meant to imply design; any solution that provides the required functionality is acceptable [000000.c.0001] <<000110.005>>.
111. <c0000> 3.1. The System will provide the following major functional capabilities: a. Subject Search,. [000000.c.0001] <<000111.005>>.
112. <c0000> 3.1. The System will provide the following major functional capabilities: b. Ad hoc Subject Search, [000000.c.0001] <<000112.005>>.
113. <c0000> 3.1. The System will provide the following major functional capabilities: c. File Maintenance, [000000.c.0001] <<000113.005>>.
114. <c0000> 3.1. The System will provide the following major functional capabilities: d. Response Generation, and [000000.c.0001] <<000114.005>>.
115. <c0000> 3.1. The System will provide the following major functional capabilities: e. Photo Storage and Retrieval [000000.c.0001] <<000115.005>>.

Figure 5. Example of Generated Specification

3.4 Change Management

To manage the changes, this process maintaining three change history files that includes RFC index (RI), system number history (SH) and version number history (VH). RFC index indicate associated RFC number to a system number. The system number history file is updated when there are any changes such as modified or new statements. The version number history file indicates the tree structure of specification version changes.

For example, from the figure 5, the RFC number 0215 indicates that the requirements statement whoops system number is <<000110.005>> need to be changed and new statement is added as depicted in Figure 6.

114. <c0000> 3.1. The System will provide the following major functional capabilities: d. Response Generation, *Error Report* and [000000.c.0001] <<000110.005>>.
3.1. The System will provide the following major functional capabilities: f. Error Recovery

Figure 6. Example RFC

The RI, SH, and VH index files are indicated in Figure 7.

[2] SC example will be discussed in 3.5 Change management

<Version Number History Index>
c0000 c0001
<System Number History Index>
00110 00110
00111 00111
00112 00112
00113 00113
00114 00988
00115 00115
- 00989
<RFC Index>
00988 0215
00989 0215

Figure 7. Example RI, SH, and VH index files

With these index files, new version of specification can be generated as shown in Figure 8.

110. <c0001> 3.1. The allocation of requirements to different functional areas is not meant to imply design; any solution that provides the required functionality is acceptable [000000.c.0001] <<000110.005>>.
111. <c0001> 3.1. The System will provide the following major functional capabilities: a. Subject Search,. [000000.c.0001] <<000111.005>>.
112. <c0001> 3.1. The System will provide the following major functional capabilities: b. Ad hoc Subject Search, [000000.c.0001] <<000112.005>>.
113. <c0001> 3.1. The System will provide the following major functional capabilities: c. File Maintenance, [000000.c.0001] <<000113.005>>.
114. <c0001> <C0215> 3.1. The System will provide the following major functional capabilities: d. Error Report, and [000000.c.0001] <<000114.005>> <<000988.005>>.
115. <c0000> 3.1. The System will provide the following major functional capabilities: e. Photo Storage and Retrieval [000000.c.0001] <<000115.005>>.
116. <c0001> <C0215> 3.1. The System will provide the following major functional capabilities: f. Error Recovery [000000.c.0001] <<000989.005>>.

Figure 8. New version of Specification

Request For Change (RFC) that is RFC 0215 in this example indicates that striped words need to be deleted and italic words nee to be inserted. After incorporate the RFC, the system specification element got new change number that is ***<c0001> <C0215>*** and system number that is <<000988.005>>. The system number is one higher number of the biggest system number in the document.

5. Application

A large software system development that we involved has two high level system description documents that include 1172 and 1474 system specification elements that are in natural language statements form, one system level description document that includes 887 system elements, and four segment level system description documents that include 684, 1206, 2922, and 1877 system specification elements. Figure 9 describes system specification documents structures and number of statements in each document. During two years period, there has been more than 900 changes are made on these system description documents.

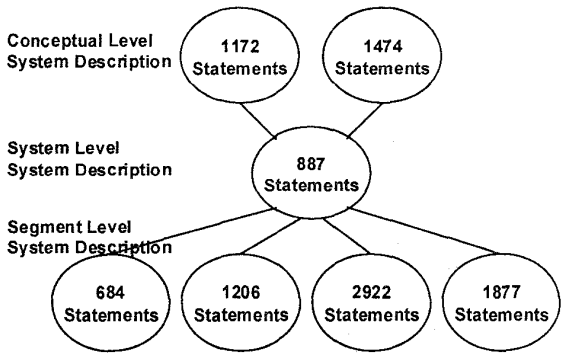

Figure 9. system specification documents structures and number of statements

By application of engineering baseline process, we could get following experience;

- The engineering baselines enables all engineers access dynamically updated requirements specification.
- The system number increase auditablity in configuration management since a typical 3.4.2.2 numbering system can not be machine audited: a machine process knows that 6 follows 5 but it does not know that, in particular case, 2.3 follows 2.2.11.
- The engineering tag index files enables any and all engineers involved in various activities such as traceability establishment, test plan developments. There are some tools available to maintain traceability. The key is to facilitate all the engineers who have domain experties. Usually they do not have specialty on these tools. The plain ASCII two column index file enables their participation by using plain word processor. To maintain engineers feedback these tools can be used as a repository of those two-column indices.

The history of system number changes indicates requirements maturity and stability that can measured as requirements volatility [CL95].

6. Conclusion

A new paradigm for use in system development was presented: engineering baselines. This paradigm enables, because of its simplicity and feasibility for implementation by everyone involved, controlled visibility and traceability into the complete development effort. Without feasibility, without an approach that all engineers can readily apply at their desks or benches, any system or process, no matter how good otherwise, eventually fails. The primary characteristics of engineering baselines include:

- All system descriptions are established in plain ASCII files, with each system description element as a standalone entity, bulleted text concatenated with lead in information etc.
- Application of three types of numbers provide controlled visibility and traceability into all aspects and dimensions of the development effort: (1) a unique system number for all system elements, (2) an engineering tag to establish links among system elements, and (3) change set numbers. These three types of numbers enable the tracking and linking of all requirements statements, including reuse. Requirements statements are linked to all type of derived/allocated requirements statements as they are created. These methods provide significantly more ability to allocate and trace requirements statements than most current practice [Dav93].
- Use of engineering tags, in conduction with plain two-column ASCII index files for all associated among requirements statements, enables comprehensive traceability among all system elements.
- A needed "engineering-based" system description structure, in a plain machine-processable ASCII file, is enabled. The typical 3.4.6.1 structure, as well as the somewhat standard "shall" type approach, does not enable the needed engineering structuring for system development. In some cases, paragraphs are too aggregated, with up to as many as 30 individual elements that need to be separately allocated and traced. In other cases, sets of "shall" may need to be aggregated into one engineering-based system element. In a third case, compound sentences, including those with a single "shall", may need to be decomposed into separate system elements. Separately controlled, but not that is often too limited. Examples of other essential non- "shall" text include those with: "s" verbs such as, transmits, receives, stores, displays; "must"; "be" verbs such as is, are, were, was; and references.
- Reuse is facilitated by links among all types of system elements, from top level specifications all the way to detailed software and hardware components. That facilitated is because the links are readily established by the engineers as they create system elements; and that is a prime criteria for their ever really getting created/documented.
- Change data are explicitly established and recorded in individual machine-processable files and associated two-column ASCII index files, for each system elements. The change page approach of the existing system development paradigm do not have change identifiers embedded in the descriptions of each system elements, when they leave a hard copy page and go into machine for processing, the change record is lost.

References

[CL95] R. Costello and D. Liu, "Metrics for Requirements Engineering," *Journal of Systems Software*, Vol 29. pp 39 - 63, 1995.

[Dav93] A. Davis, *Software Requirements: Objects, Functions, and States*, Prentice-Hall 1993.

[McD91] J. McDermid, *Software Engineering Reference Book*, Butterworth-Heinemann Ltd 1991.

[MRP95] M. Merriman, R. Evans, and S. Park, "Automated Support for Text-Based System Assessment," *Proceedings of International Symposium and Workshop on Systems Engineering of Computer-Based Systems*, March 1995.

[Ols93] N. Olsen, "The Software Rush Hour," *IEEE Software*, Sept. 1993, pp 29-37.

[RPM95] R. Evans, S. Park, and M. Merrima, "Engineering Baseline in System Development: Using ASCII files, two-column Index File, and System Numbers, Engineering Tags, and Change Set Numbers," *Proceedings of International Conference on Engineering of Complex Computer Systems*, November 1995.

[Yeh82] R. Yeh, "Requirements Analysis - A Management Perspective," *Proceedings of International Computer Software and Applications Conference* 1982, pp 410-416.