Cover letter for IRMA 2007 conference
Track submitted: Project management and IT

Author 1: James L. Goldman
Email: jimg@drexel.edu
Affiliation: College of Information Science and Technology, Drexel University
Address: iSchool at Drexel University, 3141 Chestnut St Philadelphia 19104- 2875
Ph: (215) 895-5912
Fax: (215) 895-2494

Author 2: George Abraham*
Email: sga72@drexel.edu
Affiliation: College of Information Science and Technology, Drexel University
Address: iSchool at Drexel University, 3141 Chestnut St Philadelphia 19104- 2875
Ph: (215) 895-5912
Fax: (215) 895-2494

Author 3: Il-Yeol Song, PhD
Email: sga72@drexel.edu
Affiliation: College of Information Science and Technology, Drexel University
Address: iSchool at Drexel University, 3141 Chestnut St Philadelphia 19104- 2875
Ph: (215) 895-2489
Fax: (215) 895-2494

* corresponding author

Generating Software Requirements Specification (IEEE- Std. 830-1998) **document with Use Cases**

James L. Goldman, George Abraham*, and Il-Yeol Song
College of Information Science and Technology (iSchool at Drexel)
Drexel University, Philadelphia, PA-19104
{jimg, sga72, song}@drexel.edu
* corresponding author

Abstract: The IEEE Std.830-1998 was created to standardize the software requirements specification document. The aim of an SRS document is to capture requirements in an unambiguous manner in order to facilitate communication between stakeholders. The Use Case approach has become a de-facto standard for capturing functional requirements. IEEE Std.830-1998 provides a structure (template) for documenting the software requirements. But, it does not show how to leverage the information already captured in Use Cases for generating the specification document. In this paper, we present an approach to prepare SRS with Use-Cases. We do this by employing classification schemes (use-case taxonomy) identified to manage the Use Cases. Our method provides additional support to analysts in preparing a standards compliant SRS document by avoiding redundant specification effort and through reduction in the cognitive load. We demonstrate how this taxonomy is used to develop a standards compliant SRS document with the help of a case study.

1 Introduction

The SRS document described in IEEE Std.830 is divided into a number of recommended sections to ensure that information relevant to stakeholders is captured. This specification document serves as a reference point during the development process and captures requirements that need to be met by the software product. Basic issues addressed in the SRS include functionality, external interfaces, performance requirements, attributes and design constraints. It also serves as a contract between the supplier and customer with respect to what the final product would provide and help achieve. Although the IEEE Std. 830-1998 specifies the structure it does not choose one representation for requirements over the other. Neither does it specify what techniques should be used to populate the various sections.

The Use-Case approach has become the de-facto standard for capturing functional requirements. Many of the sections of the SRS document contain information that would be otherwise collected in UML use case artifacts. A significant amount of effort could be spared if the description of functionality captured in these Use-Case artifacts is used to populate relevant SRS sections. For large projects, the number of use cases and the amount of related documentation could quickly become unwieldy without the presence of an organization scheme. It is possible to systematically create and populate several of the SRS document sections if Use-Cases are documented using appropriate organization schemes. The advantage of systematic translation is avoiding duplicative specification efforts. After all, if time and effort have been expended creating the use case artifacts, it makes sense to reuse the results of those efforts when writing the SRS document. It would also lessen the possibility of introducing inconsistencies that arise during duplication.

Presently, there are no concrete techniques to identify and link use-cases to sections of the SRS. This process is at best ad-hoc, which generates inconsistencies in the final specification document. In this paper we show a systematic way to leverage existing/discovered Use-Cases to populate the SRS document. We do this with the help of various schemes for managing and

organizing the Use Cases, and by linking specific use case types to related SRS sections. Our method provides additional support to analysts in preparing a standards compliant SRS document by avoiding redundant specification effort and through reduction in the cognitive load. We demonstrate how this taxonomy is used to develop a standards compliant SRS document with the help of case study.

In Section 2 we provide an overview of the various organization schemes used in our method for managing and organizing Use Cases. Section 3 discusses how specific UML artifacts could be linked to the content requested in specific SRS sections. In Section 4 we demonstrate how the organization scheme has been applied in a case study for developing an SRS, and Section 5 concludes our paper with discussion of future work.

2 SRS and Use Case taxonomy

The use-case model is an interpretation of the SRS (Spence & Probasco, 2000). For ease of documentation, the use-case model along with the supplementary specifications document is used as the formal documentation for the project at times. This may seem like an efficient system but it cannot be substituted for a formal SRS. The need for an SRS document is usually mandated by the management. Under such circumstances, when an SRS standards document is unavailable, the use-case model is dissected and the use case descriptions cannibalized in an attempt to populate the SRS. This process tends to be ad-hoc giving rise to inconsistencies in the final document. It also surfaces traceability issues between the use-case model and sections of the SRS document. Changes in functional requirements in the specification document need to be reflected in the use-case model and vice-versa. We should also point out that the use-case model is an abstraction of the system model. It does not capture all the relevant aspects of the system, especially non-functional requirements, which are required for completing the product documentation. An unstructured process for using use-cases to populate an SRS is inefficient and lacks traceability. The SRS forms the basis for testing plans at a later stage, further boosting its importance in software development process.

There is an incentive to prepare the SRS in accordance to the standards. It ensures readability of the document by other stakeholders who come on board at a later date. The IEEE Std.830 is understood across organizations facilitating communication between disparate organizations. It also makes sense from Information systems maintenance or systems testing perspective, where convention is preferred over unique formats unless extra-ordinary circumstances exist. At the same time there is also an incentive to avoid duplication effort.

Goldman and Song (2005) reviewed four possible schemes, and proposed one of their own, for organizing and managing use cases. The choice of schemes for managing the use cases depends on the context. The five classification schemes presented were:

a) Business Use Case vs. System Use case
b) Essential Use Case vs. Real Use Case
c) Based on Organizational Goals: Core vs. Administrative vs. Routine Use Cases
d) Based on Importance Level: Primary vs. Secondary vs. optional Use Cases
e) By Function type- 1.Data Entry/Maintenance, 2.Transaction recording, 3.Calculation, 4.Transformation, 5.Communication, 6.Device Control, 7.System Administration.

The schemes could be used either for organizing use-cases sequentially, like for e.g. table of contents, or for grouping related use-cases based on shared attributes or behavior. For a detailed discussion on this we would refer readers to Goldman and Song (2005). For our method we employ organization schemes c, d, and e.

What follows is a proposed systematic guide to the translation of software requirements specifications from UML use case models into the IEEE 830 recommended format. The translation makes use of the multiple use case classifications presented earlier in Section 2.

3 Sections of the SRS Document

The table below gives an overview of how to form each SRS document section from the appropriate information captured in the use case artifacts.

Table 1: Forming the IEEE 830 SRS Document from Use Cases

IEEE 830-1998 Section **2** of the SRS	**UML Use Cases**
Product perspective	Use Case Diagram; Component and Deployment Diagrams
Product functions	Functional requirements use cases organized by the generic type of functionality provided
User characteristics	List each actor Û use case pair with brief explanation how that actor interacts with that use case.
Constraints	Non-functional goal-oriented use cases
Assumptions and dependencies	Interdependencies between use cases, especially between functional and non-functional use cases.
Apportioning of requirements	Use case names categorized by importance (Primary, Secondary, Optional…)
IEEE 830-1998 Section **3** of the SRS	**UML Use Cases**
External interfaces	All UML actor interactions with use cases at the system boundary. Use case descriptions (narratives).
Specific Functional requirements	Functional requirements use cases organized by the generic type of functionality provided. Use case descriptions (narratives).
Performance requirements	Non-functional goal-oriented use cases related to performance
Design constraints	Non-functional goal-oriented use cases related to design
Organization of functional requirements	Use case classification schemes outlined above

3.1 Product perspective

In Section 5.2.1 of IEEE 830 (Section 2.1 of the SRS), we are told that "this subsection of the SRS should put the product into perspective with other related products. … A block diagram showing the major components of the larger system, interconnections, and external interfaces can be helpful." A use case diagram establishes the system boundary, and should show the use cases that provide functionality to actors outside the system boundary. To some extent this captures the interfaces needed for actors external to the system to interact with the system. Other sub-systems could be represented as agents and the interaction with the system could still be captured just like for a regular actor.

3.2 Product functions

In Section 5.2.2 of IEEE 830 (Section 2.2 of the SRS), the major functions that the system will perform are described in summary form. IEEE 830 offers no guidance on how to organize descriptions of the major functions (although several suggestions on organizing the more detailed functional requirements are included).

One of the classifications mentioned above, that of primary vs. secondary vs. optional use cases, can be used to narrow down the field of the possible use cases so that only primary use cases would be described in the major function summary section. The field can further be narrowed by applying the core vs. administrative vs. routine use case classification scheme to eliminate primary administrative and routine use cases. What is left are only the primary core use cases.

3.3 User characteristics

Section 2.3 of the SRS describes "general characteristics of the intended users of the product…" In the Use Case model an actor is a role. However, IEEE 830 asks for information about the backgrounds of the intended users, including "educational level, experience, and technical expertise," and these have no corresponding collection point within the Use Case, so will have to be added separately. But, mapping roles to users may aid the discovery process.

3.4 Constraints

Constraints, included in Section 2.4 of the SRS, are "items that will limit the developer's options" (IEEE 830). Constraints are also sometimes called non-functional requirements because they are requirements that the system must meet, yet they do not provide or describe functionality that accomplishes the purpose of the system. Examples include regulatory compliance requirements, performance requirements, and compatibility with externally-specified protocols and system interfaces. Representation of non-functional requirements is topic of research but, presently it is included as business rules governing the interaction.

3.5 Assumptions and dependencies

Assumptions and dependencies (Section 2.5 of the SRS) come from several places in the use case-based specification process. Some assumptions are stated in the preconditions of the functional use cases, particularly when the preconditions refer to things external to the system, whether they are actors or external systems.

3.6 Apportioning of requirements

Section 2.6 of the SRS document should "identify requirements that may be delayed until future versions of the system." This information is identified by the primary vs. secondary vs. optional use case categorization described in Goldman and Song (2005).

3.7 Specific Requirements

Section 3 of the SRS document contains the heart of the specification of exactly what the system should do and how. Section 3 revisits some of the areas that were addressed in Section 2, but suggests that this is the appropriate place for inclusion of a higher level of detail. Therefore, essential and business use cases are not appropriate for translation into Section 3, only real system use cases are.

3.7.1 External interfaces

The external interfaces described for inclusion 3.1 are a more detailed description of the interfaces mentioned in Section 2.1 of the SRS document. The appropriate place to find the corresponding information is in the narrative description of the use case primary and alternative scenarios. These are typically described in a request-and-response format, where an actor action is followed by one or more system responses, followed by further actor actions and system responses, until the completion of the use case and the satisfaction of the requirement. The set of all actor actions, and corresponding system responses, ought to suffice as “a detailed description of all inputs into and outputs from the software system.” (IEEE 830, p. 16)

A use case called “Process Sale Transaction and Payment” might include this partial use case description of a request and response in one of its scenarios:

Actor Action	System Response
Cashier scans barcode on product box.	System displays item description and current price on point of sale terminal.

From the use case description, most of the IEEE 830 items may be extracted, and restated in the following corresponding subsections.

a) Name of item: “Cashier scans barcode on product box.”

b) Description of purpose: “System displays item description and current price on point of sale terminal.”

c) Source of input: Cashier (actor name)

d) Valid range, accuracy, and/or tolerance: as stated in preconditions

e) Units of measure: as stated in use case summary, or in the scenario narrative.

f) Timing: shown by sequence of steps in the use case scenario narrative.

g) Relationship to other inputs/outputs: The most relevant related inputs/outputs will be those that are also involved in the interactions within the same use case. Others may be separately noted.

h) Screen formats/organization: If required, these should be noted as system responses in the use case scenario narratives where appropriate. For example, a requirement for a credit card entry form could be described by this scenario:

Actor Action	System Response
Cashier indicates credit card payment is desired.	System displays empty credit card entry form in window.
Cashier swipes credit card.	System processes credit card.

i) Window formats/organization: If required, these should also be noted as system responses where appropriate.

j) Data formats: These may be noted either in actor actions or in system responses, depending on where the requirement applies. For example: “System displays customer’s zip code left justified; hyphen to appear after first five digits if nine digit zip code on file.”

k) Command formats:

l) End messages: These should appear as the last system response in the scenario narrative, and/or as described in the use case post-conditions.

3.7.2 Functions

Each functional requirement of a system has an overall description which should appear as the second item in the use case description, after the use case title. More specific information can be mapped from the use case description as follows. The analyst has the choice of whether to include the UML artifacts directly as elements within the IEEE 830 SRS document or whether to abstract from the UML artifacts the necessary information to fill in these sections.

a) Validity checks on the inputs: These should be explained in the “system response” descriptions within the use case description’s scenario narrative

b) Exact sequence of operations: The entire scenario narrative for the expected case may be used to describe an exact sequence of operations.

c) Responses to abnormal situations: Alternative scenario narratives will explain how the system must respond to abnormal situations.

d) Effect of parameters: The effect of parameters may be shown through alternative path scenarios.

e) Relationship of outputs to inputs: Use case description scenario narratives also explain the relationship of outputs to inputs since they explain exactly what the system delivers, or how the system state changes, in response to each actor action document.

3.7.3 Performance requirements

Such requirements can be documented using UML use cases if the UML is extended slightly to accommodate “goal-oriented” use cases. Currently this information has to be captured explicitly during use case development because “goal-oriented” use cases are not a formal representation in UML model.

3.7.4 Logical database requirements

Unfortunately, use cases do not specifically provide for the specification of logical databases when they are used to define system functionality. Other aspects of database requirements called for by IEEE 830, are not documented in the Use Case and have to supplemented from class diagrams.

3.7.5 Design constraints and Software System Attributes

Sections 3.4 and 3.5 of the IEEE 830 SRS document external constraints that are imposed on the system’s design and implementation. The constraints are therefore documented as requirements. These include standards and regulatory compliance, along with additional non-performance non-functional requirements. The latter include requirements having to do with reliability, availability, security, maintainability, etc.

All of these requirements are non-functional goal-oriented aspects of the system, and have to be documented explicitly as they are not inherently available in the use case.

3.8 Organizing the specific requirements

IEEE 830 explicitly concedes that there are many possible ways of organizing the requirements documentation in Section 3 of the SRS. Several possible examples are shown in the appendix.

In the above mappings, an attempt is made to preserve as much information as possible from UML use case descriptions to create the SRS document in IEEE 830 systematically, if not automatically. Most of the example organizations presented in IEEE 830 can be created from these organizational schemes mentioned in Section 2 of this paper. One possible suggestion offered is to organize the requirements by user class, which would translate as actor in the UML use case model. Another is to organize them by feature, which would suggest the use of the proposed seven generic use-case function types as a classification scheme, although a domain-specific use case classification might also be helpful for more complex systems. Yet another suggestion is to organize requirements by functional hierarchy, for which a UML system level use case diagram ought to suffice.

If all functional requirements have been detailed in use case diagrams and descriptions, non-functional requirements in supplementary specifications, and the use cases have been properly classified by the various attributes describing them, then the analyst will have collected and organized most of the information necessary to create the IEEE 830 SRS document without much further manual intervention.

4 Case study

The following fictional system's requirements, expressed as a handful of use cases, will be used to illustrate the expressive power of the use case taxonomies. For this case study, we will consider only system use cases, and not business use cases. For simplicity, we will omit consideration of the target's system's interfaces with other information processing systems.

4.1 System Overview

A periodical publisher wishes to implement a computerized information management system to enable it to manage and grow its business. This publisher publishes several magazines, each of which has it's own production schedule, advertising rates, and editorial staff. However, the publisher's magazines cover closely related fields of interest, so the subscriber lists of each of the magazines exhibit large overlap with those of the other magazines. One system is required to handle all transaction, subscriber-servicing, accounting, and fulfillment activities. The system is to handle everything that is needed to publish these magazines, except for support of the editorial (content production) processes, which is handled separately.

4.2 Organization and Classification of the Use Cases

Once the use cases have been identified (refer Table: 2, Pg.7) and briefly described, they are classified by applying heuristics (Question-Method) detailed in Goldman and Song (2005). In this case study we use three of the possible classification schemes that would serve us best for preparing the SRS. The deliverable from this exercise is a classification of a use case along three dimensions:

a) Is the use case a core, administrative or routine use case?

b) Is the use case a primary, secondary or optional use case?

c) Which generic function type describes the use case functionality?

Table 2 lists the uses cases identified for the case study along assigned. The “C/A/R” column indicates whether the use case is considered “Core,” “Administrative,” or “Routine.” The “P/S/O” column indicates whether the use case is considered “Primary,” “Secondary,” or “Optional.” The “G.F.T.” column indicates the use case’s classification within the seven generic function types (Section 2). The number in the “G.F.T.” column corresponds to the subsection within Section 2.

Table 2: Magazine Publisher Case Study Use Cases and Classifications

Use Case	**Brief Use Case Description**	**C/A/R**	**P/S/O**	**G.F.T.**
1	Add subscriber information: a new subscriber’s information is added to the database.	C	P	1
2	Add new subscription: a new subscription is created in the system.	C	P	2
3	Renew existing subscription: an existing subscription is renewed for additional time.	C	P	2
4	Print fulfillment mailing labels: labels to be attached to magazine issues for mailing.	C	P	5
5	Log in: user identifies self with login ID and password.	A	S	7
6	Add advertiser: a new advertiser’s information is added to the database.	R	P	1
7	Add advertisement: an advertisement is received from an advertiser.	C	P	1
8	Schedule advertisement (process insertion order): an advertisement is scheduled to appear in one or more magazine issues.	C	P	2
9	Close issue: total the advertising revenue from scheduled ads in this issue and prohibit further advertisement insertions.	R	S	4
10	Back up certain data.	A	O	7
11	Receive payment: payments are recorded for subscriptions or for advertisements.	C	P	2
12	Cancel subscription.	C	S	2
13	Maintain issue publication schedule: record the dates on which tasks such as closing an issue and printing mailing labels must be completed for each magazine issue.	A	S	1
14	Print circulation report: print a summary report of descriptive statistics.	R	S	5
15	The system must be *n*% reliable: This is an example of a non-functional requirement expressed as a goal-oriented use case.	n/a	P	n/a
16	Initialize the system: configure default settings and start the system application.	A	P	7

17	Print loyalty report: print a report showing customers ranked by longevity and number of magazines	R	O	5

4.3 SRS Section 3

Based on this classification, we shall arrange the use cases into the "Specific Requirements" Section 3 of the IEEE 830 SRS document. We will use as our major classification the "Core-Administrative-Routine" distinction, followed by the prioritization (Primary/Secondary/Optional). The order of these classifications could be reversed: IEEE 830, Annex A shows that any of a number of hierarchical arrangements is acceptable.

Using this method, the result is an easily produced, yet organized presentation of the use cases, well suited to guiding the design and implementation process.

4.3.1 Magazine Publisher Case Study -- Sample SRS Section 3 Organization

Core

- Primary
  - *Data entry / maintenance (4.6.1)*
    - 1. Renew existing subscription
    - 7. Add advertisement
  - *Transaction recording (4.6.2)*
    - 2. Add new subscription
    - 3. Renew existing subscription
    - 8. Schedule advertisement
    - 11. Receive Payment
  - *Communication (4.6.5)*
    - 4. Print fulfillment labels
- Secondary
  - *Transaction recording (4.6.2)*
    - 12. Cancel subscription
- Optional (Note: an Optional Core use case would be rare.)

Administrative

- Primary
  - *System administration (4.6.7)*
    - 16. Initialize the system
- Secondary
  - *Data entry / Maintenance (4.6.1)*
    - 13. Maintain issue publication schedule
  - *System administration (4.6.7)*
    - 5. Log in
- Optional
  - *System administration (4.6.7)*
    - 10. Back up certain data

Routine

Primary

Data entry / Maintenance (4.6.1)

6. Add advertiser

Secondary

Transformation (4.6.4)

9. Close issue

Communication (4.6.5):

14. Print circulation report

Optional

Communication (4.6.5)

17. Print loyalty report

Non-functional requirements

Primary

15. The system must be *n*% reliable

5 Conclusion

UML provides a language and notations for identifying, documenting, and communicating system requirements, and among these use case descriptions and diagrams are most frequently used during the requirements definition stage of a project. The SRS document prepared in compliance with the Std.830-1998 can ensure unambiguous communication between the stakeholders. The use-case model alone cannot serve as the core piece of documentation as it us only an interpretation of the SRS document. But, it can shorten the time required to generate a standards compliant document if existing use-case description could be re-used in some manner.

To help avoid duplication of effort, a method for systematic translation of UML use case descriptions into the IEEE 830 recommended format has been outlined. In order to increase the comprehensibility of what would otherwise be linear lists of dozens of use cases, we have proposed that use case descriptions be categorized on five different organization schemes.

Finally, we have presented a step-by-step method based on several of the organization schemes for forming each major requirements section of an IEEE 830-based SRS document directly from information captured in UML use case diagrams and use case descriptions, and for organizing according to IEEE 830's recommended structure. In the absence of systematic techniques for preparing a standards compliant SRS document with use-cases, our method presents itself as practical solution. Our future work involves comparing the effectiveness of our technique with traditional ad-hoc approaches. This would involve evaluating the completeness of an SRS Std.830-1998 document prepared by student groups using our method and comparing it with groups using an ad-hoc process. The groups would then be interviewed to ascertain the extent cognitive load experienced in preparing the SRS document, which would serve as another evaluation metric.

6 References

1 Goldman, J. and Song, I. (2005). "Organizing and Managing Use Cases," in the First International Workshop on Best Practices of UML, Oct. 26-28, 2005, Klagenfurt, Austria, (in *Perspectives in Conceptual Modeling*, LNCS Vol. 3770, Editor: J. Akoka, etc. Springer

Verlag, 2005), pp. 43-52
2 IEEE Computer Society. (1998). IEEE Standard 1233a-1998: IEEE Guide for Developing System Requirements Specifications. New York City: IEEE.
3 IEEE Computer Society. IEEE Standard 830-1998: IEEE Recommended Practice for Software Requirements Specifications. New York City: IEEE.
4 Spence, I. and Probasco, L. (2000). Traceability strategies for managing requirements with Use cases. Rational Software White paper.