Requirements Eng (1998) 3:107–114

Requirements Engineering

Requirement Decomposition and Traceability

David P. Kirkman
Logica UK Ltd, Bristol, UK

When producing large systems, it is important that the requirement decomposition process is performed correctly, with requirement traceability maintained. This paper explores this area, using examples from industry to illustrate the problems which can be encountered if traceability/decomposition is not performed correctly. It is hoped that this paper will assist the reader in generating a more precise definition of some general system engineering concerns such as when decomposition needs to be performed, with the requirements hierarchy extended. For the problems covered, this paper also suggests methods by which they can be alleviated or recognised once they have occurred. This additional information is primarily aimed at people who are using tools to manage their requirements, as there may be mechanisms to automate these methods. This paper presents the personal view of the author, based on his many years of experience in this area.

Keywords: Requirements decomposition; Requirements hierarchy; Requirements management; Requirements problems; Requirements traceability

1. Introduction

1.1. Overview

This paper identifies some of the different sorts of decomposition problems which need to be addressed when performing requirements decomposition. The paper does not claim to identify all problems which can be encountered, just those which, from the experience of the author, are fairly common and can have a significant impact.

Correspondence and offprint requests to: David P. Kirkman, Logica UK Ltd (Bristol Office), North Quay, Temple Back, Bristol BS1 6FL, UK. E-mail: kirkmandp@logica.com

Each decomposition problem is considered in turn, with real-life examples used to bring home the impact of the problem. The problems have purposely been generalised, to prevent individual people/projects/companies being identified. It is sufficient to say that they have all come from the author's background within the defence industry, where the size of the development team at its peak would have been between 20 and 80 people.

The conclusions, which are presented at the end of this paper, identify the basic underlying problems, as opposed to the manner in which they have been manifested, with some basic rules defined which should be adhered to when undertaking requirement decomposition if the problems identified are not to be encountered.

1.2. Scene Setting

All of the industry examples quoted have been directly experienced by the author during the period from 1986 to 1996. During 1995, the author began to get directly involved, within the company he was working for, in trying to improve/refine the system engineering process, which encapsulates the requirements engineering process. This theme has continued to the current day, although the author has since changed company, joining Logica in 1997.

These industry examples quoted would not have occurred if the correct procedures were in place and were being suitably followed. The author is used to working to company standards which are based on the following:

- European Space Agency's PSS-05 standard [1];
- BS6719 specifying user requirements for a computer-based system [2];

- IEE 830 Guide to Software Requirements Specifications [3];
- IEE 1220 Standard for Application and Management of the System Engineering Process [4];
- DOD STD 490A Military Standard Specification Practices [5].

1.3. Industry Reality

Within industry, people tend to rely on their company's procedures (where they exist and are appropriate) and their own personal experience, along with that of their peers. As company procedures define what people need to undertake, it is often assumed that these encapsulate the best practices. Unfortunately, this is not often the case, with company procedures rarely kept up to date and they often don't go down to the low level needed to specify how work should be undertaken in the engineering domain.

Any shortfall could be supplemented by relevant training courses and reading relevant literature. However, companies often believe that training is not needed in this area, as their employees are adequately skilled, and employees are not often given the spare time or the incentive to read relevant literature.

From the perspective of the author, industry appears to be gradually changing, with forward-thinking companies looking in more detail at their engineering processes, with the realisation that improvements in this area could help to avoid timescale and budget/resource overruns. This improvement may give them a winning edge in an increasingly cut-throat and competitive market.

Within the defence industry, the change in the MOD's contract strategy, from cost plus to fixed price, has also been a driving force for change/improvement. In more recent times, the need to adhere and be assessed against the SEI's Capability Maturity Model for Software [6] has also been a driving force.

1.4. Domain Area

When producing large systems, it is important that the requirement decomposition process is performed correctly, with requirement traceability maintained, a point also illustrated elsewhere [7,8].

Large systems consist of a hierarchy of subsystems, with there being a need to clearly specify each system/subsystem. As a system is decomposed into its various subsystems, it is important to similarly decompose the requirements so that they can be completely addressed by the subsystem, with the subsystem fully specified, a point also addressed by Nelsen [9].

Traceability needs to be maintained to ensure that the system and the associated subsystem meet their contractual requirements, with changes to the contractual baseline also managed.

1.5. Problems Addressed

This paper addresses the following requirement decomposition and traceability problems:

- excessive 'subsystem' decomposition;
- insufficient decomposition;
- unsourced requirements;
- excessive hierarchy;
- insufficient hierarchy;
- change management.

The decomposition and hierarchy problems are distinct, as the decomposition problems look at things from the perspective of the decomposition of the system, from system to subsystem; whereas the hierarchy problems look at things from the perspective of managing the requirements hierarchy.

1.6. Document Structure

This subsection completes Section 1, the introduction; with the conclusions presented in Section 8. In between, a section is devoted to each problem addressed by this paper, as identified in Section 1.5.

For each problem, an overview is provided, prior to looking at things from the perspective of the industry examples, with the problem manifestation and impact considered. The alleviation/identification of the problem, from a generalised perspective, is then considered. Where appropriate, relevant supplemental information is included which does not naturally fit into the other subsections.

2. Excessive 'Subsystem' Decomposition

2.1. Overview

When a specification is being produced for a system, it is important to keep focused on the purpose of the specification; with one of the key points being to clearly and unambiguously specify the system.

The purpose is not to specify the internal design of the system or to directly specify the requirements/functionality of the some of the subsystems which comprise the system.

2.2. Manifestation

I once worked on a project where a junior but very knowledgeable systems engineer tried to make his mark on the project. This resulted in him providing his detailed knowledge for his area of expertise. His knowledgeable contributions were appreciated, but they were specified as requirements in an excessive level of detail, sometimes mapping on to two levels down in the component hierarchy tree beneath the system being specified with the interim hierarchy sometimes absent.

For the project I was involved in, the specification was produced as part of a bid, which meant that the specification would be a contractual document, with all requirements needing to be proved to the customer. It was conservatively estimated that the specification produced contained 30–40% more requirements than were needed to adequately specify the system; with the excess requirements more suitable for specifying the subsystems.

2.3. Impact

The problem of specifying requirements to this level of detail, where they relate to subsystems, was two-fold.

Firstly, they introduced design-related requirements which constrained the development avenues which were open to the developers when trying to define the architecture of the system. This can be a particular problem if the specification is in an area where the technology is changing rapidly or there is a significant lead time between specification and development/delivery (i.e. technology and hence likely implementation have advanced and changed).

Secondly, all requirements which constitute the specification for the system need to be tested and proved prior to the acceptance of the system. The cost incurred in carrying out this testing relates to the number of requirements and the complexity of testing each requirement. It is also worth noting that, given its formal/contractual nature, customer acceptance testing is significantly more expensive to carry out than internal testing.

Excessive decomposition can introduce a significant number of extra requirements, which will introduce additional testing costs. There is no benefit for these additional costs if the requirements have already been proved at a subsystem level.

Testing requirements which relate to a decomposition of the system (i.e. at the level of a subsystem, or below) may be extremely difficult to carry out and prove at the system level, as it may not be visibly evident that the system satisfies the requirement.

2.4. Alleviation/Identification

When producing requirements, it is important to clearly identify the system/subsystem level to which they need to relate and keep focused on this. In general, people should not produce requirements for a system and a subsystem at the same time, as this can confuse the purpose of the exercise, with the vision lost.

Within a tool for managing requirements, providing the system/subsystem level(s) to which each requirement relates are identified, the inclusion of excessively decomposed requirements in a specification can be readily identified provided the specification is produced out of the tool.

2.5. Supplemental: Bidding

This problem was identified in the context of a bid and, before proceeding, it is worth thinking in a bit more detail about bids. In a bid, you are trying to produce a specification which addresses the customer's requirements/needs and, for the reasons already outlined, it does not want to be too detailed, it needs to be clear and concise.

However, to be able to cost the development of the system, requirement decomposition is required until you reach a level where a realistic/estimated cost can be ascertained for each requirement or group of requirements. A group of requirements could be parcelled up into some sort of procurement specification, which could be placed on a subcontractor or used to identify a suitable COTS package.

This decomposition is valuable and needs to be maintained, with the traceability managed. However, requirements at this low level of detail do not want to be included in the specification produced in response to the bid.

3. Insufficient Decomposition

3.1. Overview

When a specification is being produced for a subsystem, it is important to keep focused on the purpose of the specification; with one of the key points being to specify those requirements which the subsystem is able to satisfy in their entirety.

The purpose is not to specify those requirements from the parent system which in some way relate to the subsystem. If requirements of this ilk are provided, the people responsible for the subsystem will not be sure which aspect/elements of the requirement need to be satisfied.

3.2. Manifestation

I once worked on a project where the paranoid management kept the distinct teams working in isolation. The system-level team gave two development teams the same requirement to satisfy. The requirement was along the lines that the HCI should display the composite data from a series of messages.

As an assumption, the HCI team stated that they were only responsible for displaying data and accessing data which was stored in a format suitable for display.

As an assumption, the interface team stated that they were only responsible for decoding messages and storing the data in message format.

Both of the assumptions were given in the work-package estimate that each team had prepared in isolation, without visibility to the other team's estimate. The problem was that a lot of the data needed a complicated data transformation and consolidation/amalgamation with data stored by the system, before it could be displayed.

3.3. Impact

The problem was only found when the two teams were trying to agree the detail of the interface design between the software being developed by the two teams.

When it was identified, it required an additional team. The existing two teams mentioned were similar in size, with this additional team about 40% of the size of one of the other teams. On a project that was far outstretching the budget and timescale available, this news was not welcomed and had a significantly adverse impact.

3.4. Alleviation/Identification

This problem could have been averted and included in an original work-package estimate if the teams had been given requirements they were expected to satisfy in their entirety.

This implies that system requirements cannot necessarily be directly allocated to a subsystem, if more than one subsystem is involved in satisfying the requirement. Therefore, when producing subsystem requirements from system requirements, some sort of decomposition may be essential.

4. Unsourced Requirements

4.1. Overview

When a specification is being produced for a system, it is important to keep focused on the purpose of the specification; with one of the key points being to specify requirements which the system needs to satisfy.

In the modern, commercially aware business environment, it is important in the bid stage to specify and cost a system which meets the needs of the customer and satisfies the outline specification supplied with the invitation to tender.

It is extremely easy to specify what it is believed that the customer needs which may not actually be what he wants. This is an area where a lot of engineers are prone to stray and over-specify the system; with functionality specified which may be quite complicated, with this being above and beyond the needs of the customer. In short, some engineers tend to over-engineer the solution, by trying to provide an idealistic solution, rather than a pragmatic solution which falls within the customer's budget. The issue of 'anonymous' requirements is also addressed elsewhere [10,11].

4.2. Manifestation

I once worked on a project where a systems engineer had previous experience of similar technology, but from a different domain/business area, where the environment was a lot more demanding. This resulted in him providing his detailed knowledge for his area of expertise, but from a perspective different from that of the customer. His knowledgeable contributions were appreciated, but when they were specified as requirements no traceability could be found to the customer's needs.

4.3. Impact

The project carefully considered the potential impact of these unsourced requirements and took the sensible decision to introduce them, where appropriate, in the bid as costed options. If they had been included in the basic solution, the cost would have far exceeded the customer's available budget.

4.4. Alleviation/Identification

The basic rule is that all top-level requirements in a requirements hierarchy must come from an approved

source. Unsourced requirements should be rejected or marked in some way as 'optional extras'.

When using a tool, if the correct relationships have been established, it is relatively easy to check that the top requirement in a requirements hierarchy is linked to an approved source, with new/additional sources needing the approval of the chief systems design authority before they are incorporated into the requirements database.

4.5. Supplemental: Design Requirements

When producing subsystem specifications, it is recognised that not all requirements can be traced to the specification for the parent system; some design requirements may need to be introduced to ensure that the subsystems correctly hang together or to ensure that the functionality of the subsystem is fully specified.

To allow for this, new/unsourced requirements can be introduced into the requirements database, but only after some sort of formal review, with this review acting as the reference point for these requirements coming from a newly approved source.

This provides a control mechanism for the chief design authority, with only approved new 'source' requirements introduced into the master requirements database.

5. Excessive Hierarchy

5.1. Overview

When there is a need to decompose requirements, it is important to keep focused on the purpose of the exercise and to ensure that not too many intermediate hierarchy levels are introduced prior to meeting the objective of the decomposition.

The purpose is not to see how large a hierarchy can be generated by the decomposition.

5.2. Manifestation

I once worked on a project where the customer's original requirement statements were very poor, with multiple requirements contained in a single statement. Decomposition was required to produce individual/atomised requirements.

However, there was one junior systems engineer on the team who progressively broke down the requirements he was responsible for, until he could break them down no more. This actually resulted in a requirements hierarchy which was five levels deep in some places, with a significant number of requirements in the database which served no useful purpose other than to show his thought process.

5.3. Impact

There is a configuration management overhead associated with managing a requirements database and this is directly related to the number of requirements in the database. Although a hierarchy is essential for traceability, an excessive hierarchy unnecessarily increases the complexity of the requirements management task.

5.4. Alleviation/Identification

It is extremely important to recognise what is 'work-in-progress' and not to baseline information until this 'work-in-progress' is complete. It is often worth stipulating that people should not increase the requirement hierarchy level by more than one during their decomposition work, as any interim levels are likely to be superfluous to the requirements database. This approach also has the added benefit of making the reviewing exercise of the requirements decomposition a lot easier.

5.5. Supplemental: Customer Education

It is also worth making one additional point here, as we need to educate our customers in good requirements engineering and requirements management. Although the frequency is reducing, we are still receiving specifications that have single statements which contain multiple requirements, which gives us an atomising/decomposition task before we can proceed with the main system engineering task.

6. Insufficient Hierarchy

6.1. Overview

When there is a need to decompose requirements, it is important to keep focused on one of the objectives of requirements management to ensure that appropriate hierarchy levels are baselined.

Requirement management tools allow a requirements hierarchy to be constructed and it is there for a purpose: to ensure that the evolution of a requirement is fully traceable within the database. The words of a baselined requirement should never be changed; if it is, then full traceability is lost.

There are several occasions when a baseline is required, such as:

- customer requirements received;
- decomposition exercise completed, with a formal minuted review held;
- specification issued.

6.2. Manifestation

I once worked on a project where there was very poor requirements management. When one particular person was producing the specification for their subsystem, there was a system requirement very close to that which he felt his subsystem could achieve, so he physically changed the words of the requirement.

6.3. Impact

The major problem was that the requirement he changed was one that had been specified in the customer's original specification. This problem was only picked up at a very late stage, when the system acceptance tests were being discussed with the customer, with the discrepancy in the wording of the requirement noted. Needless to say, it was very embarrassing and not easy to correct at that late stage.

6.4. Alleviation/Identification

With requirements management, one of the key things to ascertain is when a baseline needs to be established, with there being a minimal hierarchy between baselines. No one, except the chief design authority, should have the right to physically change the wording of baselined requirements. Another trick is to assign ownership of source/customer requirements to a fictitious 'customer' user.

This section has considered an insufficient hierarchy; the previous section addressed an excessive hierarchy. There is a happy medium somewhere.

7. Change Management

7.1. Overview

When managing requirements, it is important to properly manage changes in the requirements themselves. From a requirements management perspective, the purpose of the exercise is to provide traceability from the original statement of the requirement to the new statement of the requirement.

If a requirement is superseded, by a newly phrased version or a particular interpretation is made during the decomposition of the requirement, such as a design decision, then the decision needs to be clearly recorded. If it is not, problems can be encountered, such as those shown below.

7.2. Manifestation 1

I once worked on a project where a decision on the interpretation of a requirement was made during the early stages of the project, where the impact on the whole system was considered.

At a later stage, after some changes of personnel, the team most impacted by the change recognised the difficulty in addressing the requirement and mounted a very strong lobby to get it changed. The vocal nature of their lobbying almost held sway. The requirements database only contained the decomposition, not the reason for the decomposition, so the team felt justifiably aggrieved.

7.3. Impact 1

Before the situation reached a crisis point, someone who had been on the project during its early stages recalled a similar discussion in the early days of the project and, after a lot of searching, found an archived memo that he had received on the topic, which clearly explained why the particular decomposition route had been chosen, with the reasons still justifiable.

Due to poor management of the change process, a lot of needless time was wasted, with some goodwill being lost within the project team.

7.4. Manifestation 2

I once worked on another project where, in a telephone conversation, the project manager and the customer manager agreed to remove a particular unfeasible/difficult-to-implement requirement in exchange for the introduction of a particular needed bit of functionality.

The telephone call was never minuted or confirmed in writing, with the project manager physically deleting the requirement from the database.

7.5. Impact 2

Time went on and a new customer manager arrived, who went back to the original specification and noticed that one requirement was not being addressed, with no reason recorded as to why. Needless to say, a lot of time was wasted going through old log books and talking to the previous customer manager to get the situation resolved.

This situation falls into 'Change Management', as the original unfeasible/difficult-to-implement requirement should be linked to a requirement element called 'deleted requirement', with the reason for this recorded. Having a 'deleted requirement' element has the advantage that it is readily comprehensible why that particular requirement has failed any consistency checks placed on the requirements database.

7.6. Alleviation/Identification

For both of the above manifestations, the original requirement should have been part of a baseline, with any changes to it formally recorded and needing appropriate project approval. The change management process needs to address physical changes to the requirement and any changes to the interpretation of the requirement, including selecting a particular interpretation.

For proper change management, the basic process involves someone raising a fault/problem/observation/issue with a requirement, with the way the problem manifests itself recorded. Some analysis then needs to be done, to determine what the actual/underlying problem is. This analysis should include what possible solution alternatives were considered, with linkages to any source which documents the decision making process.

Change management is a legitimate reason for providing an additional level in the requirements hierarchy. In addition, if the original requirement already had a hierarchy beneath it, some analysis will need to be done of those requirements which were previously part of this hierarchy to see if they are still applicable and to assess the likely impact of any change.

8. Conclusion

8.1. Overview

Requirements decomposition, through a requirements hierarchy, is an essential part of the requirements management process and is an activity which needs to be undertaken.

To ensure that it is done properly, steps need to be taken to address various decomposition problems. The basic solution is to introduce a requirements management process.

8.2. Process Fundamentals

The basic fundamentals of any requirements management process are as follows:

- originating/new requirements must come from approved sources;
- reasons for extending the requirements hierarchy need to be clearly identified;
- relevant requirements baselines need to be established;
- baselined requirements can only be changed if the decision is recorded; this change needs appropriate authorisation (e.g. chief design authority). (Note: this also relates to taking a particular interpretation of a requirement.)

8.3. Hierarchy Extensions

The reasons for extending the requirements hierarchy need to be clearly identified and, as a rule of thumb for any particular activity, no more than one level should be added to the requirements hierarchy.

Whenever an extension to the requirements hierarchy is made, a review needs to be carried out to ensure that the activity has been completed and that the decomposition process has been carried out correctly, with nothing overlooked/missed.

Some typical reasons for introducing another level in a requirements hierarchy are:

- atomising customer requirement statements;
- system → subsystem decomposition activity (which could be to generate a requirement that the subsystem can wholly address or for bid costing purposes);
- formally agreed changes to requirements, arising from a review/meeting.

Whenever a new level in a requirements hierarchy is approved, it needs to be baselined. Subsequent revisions must follow the change management process.

8.4. Decomposition Activity

When a requirement decomposition activity is undertaken, a clear understanding of the purpose of the decomposition needs to be established.

The activity definition needs to include both what needs to be done and what system/subsystem level the requirements need to be specified at.

Whenever a decomposition activity is completed, some sort of review needs to take place to ensure that all aspects of the starting requirements have been fully addressed by the decomposition exercise.

8.5. Tagging Requirements

It is important to associate other information with a requirement. When tagged with this information, a full picture is achieved.

This paper has identified that tagging is required to identify:

- the source from which the requirement comes (i.e. requirement must come from a source or be part of a hierarchy);
- the system/subsystem level which the requirement relates to;
- the change management/decision-making process which led to the derivation of the requirement;
- the status of a requirement, such as whether it is optional or deleted (this could be extended to denote whether it is mandatory, desirable, etc.).

In a full requirements management process, a lot more information needs to be associated with a requirement, but this falls outside the scope of this paper.

8.6. Tool Usage

Managing requirements and ensuring traceability is an important activity, with this best accomplished using some sort of tool. A tool should be selected/developed to follow the defined requirements management process and support the development of a requirements hierarchy.

Correct usage of a suitable tool may enable automatic methods to be instigated to address the alleviation/identification given in this paper. If requirements are suitably tagged, the power provided to the user through a tool is significantly enhanced.

By using tools which support requirements management, in a suitable manner, checks can be performed which can highlight whether any requirement decomposition problems have occurred.

8.7. Summary

If a mature requirements management process is defined and adopted, with appropriate checks/reviews instigated, the decomposition problems identified should not occur.

The first step along this road is to identify what the problems are and come to a realisation that they need to be addressed, given the serious impact they can have. It is hoped that this paper helps to address this aspect.

Any requirements management process will need to evolve, in light of new experiences and newly encountered problems. This evolution will probably result in the realisation that some sort of tool is required to automate some aspects of the requirements management process.

The key thing is to ensure that a requirements management process is adopted which prevents the requirements decomposition and traceability problems identified in this paper from occurring.

References

1. Mazza C, Fairclough J, Melton B, De Pablo D, Scheffer A, Stevens R. Software engineering standards. Prentice-Hall, Englewood Cliffs, NJ, 1994 (earlier version in 1987)
2. BS6719. Specifying user requirements for a computer based system, 1986
3. IEE 830. Guide to software requirements specifications, 1984
4. IEE 1220. Standard for application and management of the system engineering process, Issue 1
5. DOD STD 490A. Military standard specification practices, Issue 1
6. Paulk M, Curtis W, Chrissis M, Weber C. Maturity model for software, Version 1.1, CMU/SEI-93-TR-24. Software Engineering Institute, Pittsburg, PA, 1993
7. Davis A. Tracing: a simple necessity neglected. IEEE Software 1995;12(5)
8. Ramesh B, Powers T, Stubbs C. Implementing requirements traceability: a case study. In: Proceedings of the 2nd IEEE international symposium on requirements engineering (RE 95), York, 1995
9. Nelsen E. System engineering and requirements allocation. In: Thayer R, Dorfman M (eds), System and software requirements engineering. IEEE Computer Society Press, Washington, DC, 1990
10. Hutchings A, Knox S. Creating products customers demand. Commun ACM 195;38(5)
11. Potts C. Invented requirements and imagined customers: requirements engineering for off-the-shelf software. In: Proceedings of the 2nd IEEE international symposium on requirements engineering (RE 95), York, 1995