Assignments and Hints

Pearson LiveLessons

- This downloadable file contains *Assignments* and *Hints*, which are referenced to the appropriate sub-lesson number.
- The purpose of an *Assignment* is to ask you some questions and set you exercises to help you explore and reinforce the subjects covered by a particular sub-lesson.
- The purpose of *Hints* is to remind you of some techniques that are particularly useful in considering how to apply the ideas covered by a particular sub-lesson.
- This material also contains a glossary of terms used in the video course.
- The Requirements Knowledge Model is reproduced in full.
- The table of contents for the Volere Requirements Specification Template is also reproduced.
- We should also mention our book, *Mastering the Requirements Process – Getting Requirements Right,* 3rd edition, published by Addison Wesley.
- You can read articles about requirements at www.volere.co.uk
- We have a discussion forum at the Volere Requirements LinkedIn grouphttp://goo.gl/NHGwT

1.1 Assignment: Establish your requirements process

Consider the following questions and mentally answer them. These are here to reinforce the sub-lesson about requirements processes.

1. What is a requirements process?
2. Why do you need a requirements process?
3. Why do requirements processes vary from project to project?

2.1 Assignment: Lay the foundations

1. What are the three elements that you want to get into balance?
2. What is the difference between the scope of the work and the scope of the product?
3. How will you identify the appropriate stakeholders for the work you need to study?
4. How do you know if you have a well-defined goal?
5. What can you count at the end of your blastoff to help make decisions about the project?

2.2 Assignment: Scope the problem

1. What is a rich picture?
2. Sketch a rich picture to show all the elements you consider relevant to cooking a meal.
3. Try another rich picture to show all the elements for planning a holiday.

2.3 Assignment: Context diagram

1. What are the 3 components of a work context diagram?
2. What is the purpose of drawing a work context diagram?
3. Try sketching a work context diagram for a piece of your work.
4. How would you use a work context diagram to communicate with other people?
5. What advantages do you get from setting the scope of the work?
6. What problems do you foresee?

3.1 Assignment: Event partitioning

1. Choose a piece of work with which you are familiar.
2. Identify a business event: The name of the event, the input, the output(s).
3. Define the data content of the input and output data flows for your event.
4. Write a one-sentence summary of the business use case (the work's response to the event).
5. Add the inputs and outputs to a context diagram and then return to point 1 and identify another business event.
6. Identify as many business events as you can and add each one to your context diagram.

3.2 Assignment: Listen to stakeholders

1. What is the difference between hearing and listening? Try to explain this difference to a colleague.
2. Why do you need a variety of techniques for discovering the requirements?
3. What is the difference between the two techniques: apprenticing and interviewing?

3.3 Hints: Discovery techniques

- Use the data to guide you to ask relevant questions.
- Use a variety of trawling techniques to elicit, discover and invent requirements.
- Record everything. Use a notebook, whiteboard, camera, phone, etc.
- The dictionary is the basis for formalising your requirements.
- When you explore the problem it is likely that you will discover changes in the work context, treat this as a success.
- If someone tells you "it is obvious" be sure to write it down, it won't be obvious to everyone.

3.4 Assignment: Data dictionary

1. How does the data help you explore the problem?
2. When would you start your data dictionary?
3. Who is responsible for defining the data?
4. Why is a simple notation recommended for the data dictionary?
5. How can you improve data definitions in your projects?

3.9 Assignment: Get to the essence

1. What are the four viewpoints identified by the brown cow model?
2. What is the benefit of getting to the essence?
3. What is a key skill to enable you to discover the essence?

4.1 Hints: Generate ideas

Innovation Triggers provide inspiration for innovation. Think of it as a guide to some facet of your business that needs an innovation. One problem with the triggers is that they sound too general, and perhaps too obvious.

Well, they are obvious, but they belong to that category of things that are obvious once they are pointed out.

We humans are not good at having ideas without some kind of aid, but once given a direction, a problem to solve or some other influence, we are very good at having ideas. In other words, we are better at improving things than we are at coming up with something from scratch.

The innovation triggers are:

- Information
- Participation
- Speed
- Convenience
- Differentiation

And some others we did not talk about in the course:

- Service
- Connections
- Green
- Responsible

Information: Your customers already have lots of information, and expect more of it. Moreover, not giving information is often taken to mean that you (the organization not providing the information) have something to hide.

People need information to make their decisions. For example, Amazon is a successful seller of a wide range of products, and part of this succe2ss is down to the amount of information that Amazon provides about its products. Additionally, customer reviews provide useful information. On numerous occasions I have bought an alternative product to the one I first looked at because of unfavorable customer reviews. So even bad news is good information.

Think about the eventual users of your product. Spend some time thinking about the information that if provided, would make it easier or more convenient to do whatever task they use your product for.

Participation: People want to be more involved in their transactions, and do more of the work themselves. We go online to buy things and arrange their delivery, we book our flights, download our boarding passes to our phones, check ourselves out of supermarkets, track our parcels, and do much of the work that used to be done for us.

This is participating in the business, and we seem to like it. The attitude is that by doing things ourselves, we are not dependent on others, and whatever it is that we want, we will get it more conveniently and more quickly, if we do it ourselves.

The question that you, the business analyst, must ask is, "What can we do to have our customers participate more in our business?" "What facilities can we provide that allow customers to become more a part of our business?"

Speed. We have become accustomed to things being done quickly. And despite things being faster now than they were yesterday, we want them to be even faster. For example, how have your expectations about speed changed over the last five years? Probably quite considerably.

We want things to happen more quickly. This means that whatever system or service you are building, it must make the customer or user think that they are getting something quickly. It does not actually have to be faster, but the end user must think he/she is getting something more quickly than before.

So think about speed: what is it that your product or system can do to be faster in the eyes of its users? Can you change a process to eliminate a few steps and get something to the customer more quickly? Can you change the product so that the customer does some of the work himself? By doing the work themselves, customers usually *think* it is faster Think about speed – people love it.

Convenience is probably the important of the innovation triggers. We value convenience, and are prepared to pay for it. I know it sounds obvious, but our systems are not always as convenient as they can be, and the more convenient they are, the more likely they are to be used, and the more people will use them.

Consider recorded music, and how it has changed over the years. The vinyl album gave way to cassettes, these were overtaken by the compact disc, which in turn is being supplanted by streamed music. These were not

necessarily advances in the quality of the music (good vinyl on a high-quality turntable still outperforms them all) but each was more convenient than its predecessor. We bank online because it is more convenient than going to the bank and standing in line for the teller. Home delivery is more convenient than going to the store and buying it oneself. The newspaper on the iPad is more convenient than the paper version bought at the corner kiosk.

You have to ask, "Is this product / service / system as convenient as I can make it?" Is there something – anything—that you can do to make it more convenient for its user? Convenience attracts customers and users. Spend five minutes and generate an idea that makes your product more convenient.

Differentiation is not the same as being different. Differentiation is the separation of your product from your competitions'. You can make a telephone different by painting it yellow, but to differentiate your telephone from other telephones you need to change it so that it gives you directions when you are walking around an unknown city. The differentiation is important when you consider consumers who will have to switch from whatever they have at the moment. The differentiation must overcome their reluctance to switch, or it must provide enough of a benefit to overcome the dominant design.

It is useful to keep in mind that the greater the differentiation, the greater the benefit to you. Producing a "me too" product rarely means success. Producing the breakthrough product – the iPod, the flat screen, the Internet, mobile payments, etc. – almost always results in a runaway success.

So consider the system you are developing. Is it merely following the herd? Or can you make it one that will lead the herd in a new direction?

Service means understanding the business you are in, and thereby knowing what your customers want from it. Each day, companies go out of business because they do not provide the service that their customers want. Each day, new companies are successful because they have understood what their customers really want. This is service at its most basic.

Think of service from your customers' viewpoint. What are they trying to do, and what could your organization do to make it easier for them? What extra service can you provide that your customers will value?

Think particularly about the help and support your customers need. Think about making something more convenient (mentioned above). Think about

yourself as a customer or user of the system you are building. What service can it provide to make your life perfect?

You can use **Connections** as a trigger for innovation. We love to be connected – note the popularity of mobile phones, Facebook, and other social networks, messaging apps, and the annoying people who walk while texting. People crash their cars to answer their phones, and step into the path of oncoming traffic while checking messages; such is the need to be connected. (By the way, I am not exaggerating here. People have really been killed by their blind desire to stay connected.)

Coming back to your systems and services, your customers measure you by the way you respond to them: how you answer their questions, how you support them, how you keep them informed of your products and services. This response is the active part of connecting to the customer.

Green. People want to be green, or at least, they want to feel green or seen to be green. This often translates into wanting to buy from companies, or use services or products they feel are green, or have been produced in a green way.

Green pays—many companies have ridden to success on the green credentials, or they are currently adopting green polices. Look at how British Petroleum, known as BP, stresses its greenness despite being an oil company. Many airlines sell customers offsets for the carbon emissions from their flight. This does not necessarily bring in a flood of new customers, but it might well retain customers who want to be green. And let's face it, with so much attention (finally) being paid to the environment, being green is great advertising.

Responsible following on from greenness is responsibility. Companies can advertise their civic responsibilities and become attractive to the same audience that is attracted by greenness. Responsibility might have the added benefit that it could well be cheaper to be responsible than it is to be green.

The intention of using it as a trigger is that organizations can innovate products and services that display a civic responsibility. Ben&Jerry's Ice Cream was one of the pioneers at this. It bought them a lot of custom from the socially aware.

The simple (but not simplistic) question is, "How can my organization appear more responsible?"

The innovation triggers seem obvious. Yet it is remarkable how many projects forget these things when it is so easy to run through the list of triggers and for each one ask, "What can we do to provide better information / speed / convenience / etc.?"

4.3 Assignment: Understand the people

1. What is the difference between the two views: "What Now" and "Future What"?
2. What is the benefit of inventing a persona?
3. Work with a colleague to invent a persona who you think would be helpful in your environment.

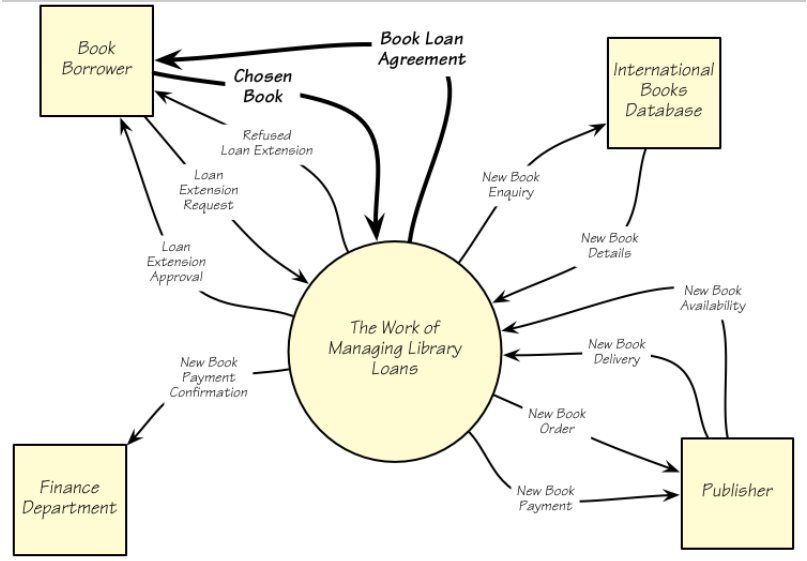
4.4 Assignment: Define future work

1. What are the different skills of a business analyst and a user experience designer?
2. What is the difference between information design and visual design?

4.5 Hints: Choose technological solution

- There is always more than one possible solution, usually different people have their own favourite – there is no one right answer.
- You need to make the different solutions visible so that you can compare them and make a choice.
- Project constraints are helpful in guiding solution choices.
- Keep track of why you have chosen a particular solution – design rationale. This will help you to make future changes and help new people understand the design.

5.3 Assignment: Functional requirements



Book Loan Agreement =

Borrower Name + Borrower Id + Library Id + Book ISBN +

Loan Start Date + Loan Expiry Date

Chosen Book = Borrower Id + Book ISBN

1. Use the definitions given for ***Chosen Book*** and ***Book Loan Agreement*** and write some functional requirements.
2. For each requirement, write *Requirement ID*, *Description*, *Rationale* and *Fit Criterion o*n a Volere snow card (included in downloads).

5.4 Assignment 1: Non-functional requirements

1. Consider a project from your own work
2. Rank the non-functional requirements in order of importance for your project and organization. Importance is business need or potential for failure if missed.

3. Write a non-functional requirement for each of your top three categories. Include requirement type, description and rationale.

NON-FUNCTIONAL REQUIREMENTS

10. Look and Feel Requirements
11. Usability Requirements
12. Performance Requirements
13. Operational Requirements
14. Maintainability Requirements
15. Security Requirements
16. Cultural and Political
17. Compliance Requirements

5.4 Assignment 2: Non-functional requirements

1. Why are non-functional requirements important?
2. Who are the sources of non-functional requirements?
3. How do you discover the non-functional requirements?
4. Which type/s of non-functional requirements are the most difficult to discover – and why?
5. What are the non-functional requirements for a bathroom tap?

5.4 Hints: Non-functional requirements

- Use the Volere table of contents checklist to identify non-functional requirements to support specific functional requirements.
- Write requirement description to summarize the intention of the requirement.
- Write the rationale to explain why the requirement is important.
- Identify solutions disguised as requirements by analysing the rationale.
- Write the fit criterion to make the requirement testable.
- Add definitions of new terms to your data dictionary.

- Involve only the stakeholders appropriate for the type of requirement.

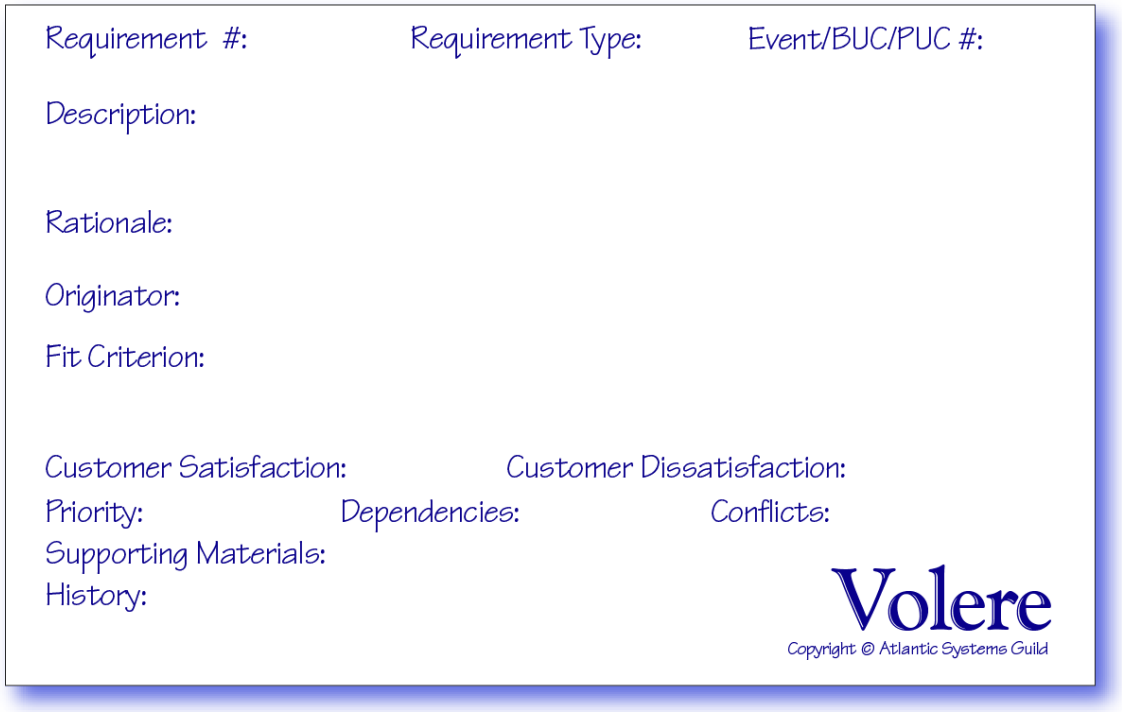
5.5 Assignment: Atomic requirements

1. What does it mean to say a requirement is “atomic”?
2. Why is it useful to have different types of requirements?
3. Do you always have requirements of every type?
4. Why do you identify the functional requirements before the non-functional requirements?
5. What is the reason for the requirement rationale?

5.5 Hints: Atomic requirements

- Keep your own list of verbs and use them consistently within your project.
- When writing requirements use exactly the same data names as the ones you have on the context diagram and in the data dictionary.
- Give each requirement a unique identifier.
- Write a description, rationale and fit criterion for each requirement.
- Make sure that the terms you use in each requirement have been defined in your data dictionary.
- Use the **Volere snow card** as a checklist for writing your atomic requirements.

5.5 Hints: Volere snow card



6.2 Assignment: Discover relevant deviations

Consider the following scenario for checking a passenger onto a flight. Identify the exceptions and alternatives that you consider most relevant in this situation:

1. Get the passenger's ticket or record locator.
2. Check this is the correct passenger, flight and destination.
3. Check the passport is valid and belongs to the passenger.
4. Record the frequent flyer number.
5. Find a seat.
6. Ask security questions.
7. Check the baggage onto the flight.
8. Provide boarding pass and bag tags.

7.1 Assignment: Making requirements testable

1. Why have a fit criterion as part of an atomic requirement?
2. Who do you think should write the fit criteria?
3. What advice would you give another business analyst about how to write fit criteria?
4. Who are all the people who use the fit criteria?
5. Why does an atomic requirement have more than one attribute?
6. How is the Quality Gateway implemented in your project(s)?

8.1 Assignment: Requirements strategy

1. How can you ensure that your project uses a common language for communicating requirements knowledge?
2. What are the variables that influence your requirements strategy?

Glossary of the terms used in this course

Adjacent system — a system that provides information to, or receives information from, the work. We study the adjacent system to understand why and how it communicates, as well as understanding the reason that it communicates with our work.

Agile – this term generally means developing systems using iteration, prioritization, and discovering the requirements in a "just in time" manner. This is also referred to in this course as "iterative".

Apprenticing — the business analyst learns the work or business that the end-user does by doing the work under the end-user's supervision.

Association — a business link between classes. It's also known as a **relationship**.

Attribute — an element of stored data. Also see **data element**.

Atomic Requirement — a measurable, testable requirement that does not need further decomposition. See also: Functional Requirement, Non-functional Requirement.

Brainstorming — a group of interested, bright people who try to generate as many ideas as possible for the product by feeding off one another's ideas.

Business analyst — the person in, most organizations, responsible for discovering and recording the requirements. It's also called requirements analyst, requirements engineer or systems analyst.

Business event — something that happens and causes the work to respond. Business events happen either outside the work in the adjacent systems, or because it is time for the work to produce some service. Business events are such things as *Customer pays an invoice*, *Truck reports all roads have been treated*, *Time to read electricity meters, Viewer wants to watch TV*.

Business use case (BUC) — the response that the work or business makes to a business event. Think of this as a self-contained amount of functions and data that are activated in their own discrete time frame.

Class — a collection of data attributes about a single subject relevant to the work.

Client — the person who pays for the development of the product. This can be someone in your organization, or someone outside for whom you are building the product. It's also known as Sponsor. See also customer.

Constraint — a pre-existing restriction that limits the solution you can provide. This can be a design constraint (the product shall run on a mobile telephone) or a project constraint (the product shall be available three months from now). A constraint is a type of requirement.

Context diagram — a diagram showing the work to be studied encapsulated as one process, and the data connections to the outside world represented as adjacent systems.

Context —the subject matter, people and organizations that might have an impact on the requirements for the product. The context of study, or the work context, identifies the work that is to be studied, and the adjacent systems that interact with this work. The product context identifies the scope of the product and its interactions with users and other systems.

Cooperative adjacent system — an adjacent system that provides an immediate and predictable service to the work or product. It is usually an automated system with a database containing information used by the work.

Customer — the person who buys the product. See also client.

Data element — an individual item of data that is not subdivided for requirements purposes. It's also known as **attribute**.

Data flow — data that moves from one process to another, usually represented by a named arrow.

Data store—a repository of data. It's used as a generic name for file, database, etc.

Design — the act of crafting a solution to fit the requirements.

Developer — usually refers to someone who builds the resultant system. It's also known as programmer.

Entity — see class.

Eventual owner — the person or organization that will ultimately own the developed product or system. This could be an organization that develops a system for in-house use, or a customer who buys the finished product.

Fit criterion — a quantification or measurement of the requirement so that you are able to determine if the delivered product satisfies the requirement.

Function point — a measure of functionality. This is used in requirements projects to measure the work area in order to project the amount of effort needed to study it and write the requirements.

Functional requirement — something that the product must do. Functional requirements are part of the fundamental processes or the essence of the product.

Non-functional requirement — a property, or quality that the product must have, such as an appearance, or a speed or accuracy property. Non-functional requirements specify how well the functionality must be carried out.

Product — that which you are about to build, and for which the requirements are written. Product usually means a software product, but the requirements can be for any kind of product, system or service.

Product use case (PUC) — part or all of a business use case. The product use case is an amount of functionality allocated to the product. Requirements are written for the PUC.

Project sociology analysis — the act of identifying all the stakeholders for the requirements project.

Prototype — a simulation of the product using either software prototyping tools, or lo-fidelity whiteboard or paper mock-ups. The purpose of the prototype is to aid the gathering of requirements. It is not used here to mean a design prototype.

Requirement — something that the product must do, or a property that the product must have.

Requirements analyst — see business analyst.

Requirements knowledge model — a class diagram showing the information that is typically gathered by the requirements activity.

Requirements Specification — a document that contains the requirements. The specification defines the product, and may be used as a contract to build the product.

Requirer — anyone who has requirements. This is also referred to as a **stakeholder**.

Snow card — an 8 by 5 inch card with the components of the requirement printed on it. It is used as a low-tech way of writing requirements.

Stakeholder — a person with some interest in the product. For example, the client for the development, a user, or somebody who builds the product. Some stakeholders are remote, for example an auditor, a safety inspector, a

company lawyer. Stakeholders as used here are not necessarily financial stakeholders.

Story — simple description of a capability from the point of view of the person who needs the new capability. It's used in agile development as the basis for deriving atomic requirements.

System — any combination of humans, hardware technology, software technology or materials that carries out a defined purpose. It is best qualified to explain the type of system under discussion. This could be a socio-technical system that involves people and technology, a computer system that involves a mixture of hard and soft technology, a software system that involves soft technology, or a social system that just involves human beings. System in this course never means just the computer, or the software system.

Systems analysis — the craft of modeling the system's functions and data.

Use case — this term is elastic and should be qualified as either: a business use case (BUC), product use case (PUC) or system use case (SUC).

User — the person, or system, that has a direct connection with the product. It's also known as end-user.

Volere – Volere is the name given to the collection of techniques, courses, templates, books, etc. created by Suzanne and James Robertson. More at www.volere.co.uk.

Work — the business area that eventually will include the product. The work is the part of the world that it is necessary to study in order to discover the requirements for the product. The work contains people, existing computer systems, business processes, roles, business rules and anything else that affects the decision about what to build or change, and whether it will fit into the world.

Volere Requirements Specification Template

The Volere Requirements Specification Template is intended for use as a basis for your requirements specifications. The template provides sections for each of the requirements types appropriate to today's software systems.

The following is the table of contents only. It is intended to give you an idea of the scope and depth of this d document. You may download the complete template from the Volere site www.volere.co.uk and adapt it to your requirements gathering process and requirements tool. The template can be used with Requisite, DOORS, Caliber RM, IRqA, Yonix and other popular tools. There is a review of requirements tools at www.volere.co.uk/tools.htm.

Table of Contents

3. Constraints

3a. Solution Constraints

3b. Implementation Environment of the Current System

3c. Partner or Collaborative Applications

3d. Off-the-Shelf Software

3e. Anticipated Workplace Environment

3f. Schedule Constraints

3g. Budget Constraints

3h. Enterprise Constraints

4. Naming Conventions and Terminology

4a. Glossary of All Terms, Including Acronyms, Used by Stakeholders Involved in the Project

5. Relevant Facts and Assumptions

5a. Relevant Facts

5b. Business Rules

5c. Assumptions

6. The Scope of the Work

6a. The Current Situation

6b. The Context of the Work

6c. Work Partitioning

6d. Specifying a Business Use Case (BUC)

7. Business Data Model and Data Dictionary

7a. Business Data Model

7b. Data Dictionary

8. The Scope of the Product

8a. Product Boundary

8b. Product Use Case Table

8c. Individual Product Use Cases

9. Functional Requirements

9a. Functional Requirements

Non-functional Requirements

10. Look and Feel Requirements

10a. Appearance Requirements

10b. Style Requirements

11. Usability and Humanity Requirements

11a. Ease of Use Requirements

11b. Personalization and Internationalization Requirements

11c. Learning Requirements

11d. Understandability and Politeness Requirements

11e. Accessibility Requirements

12. Performance Requirements

12a. Speed and Latency Requirements

12b. Safety-Critical Requirements

12c. Precision or Accuracy Requirements

12d. Reliability and Availability Requirements

12e. Robustness or Fault-Tolerance Requirements

12f. Capacity Requirements

12g. Scalability or Extensibility Requirements

12h. Longevity Requirements

13. Operational and Environmental Requirements

13a. Expected Physical Environment

13b. Wider Environment Requirements

13c. Requirements for Interfacing with Adjacent Systems

13d. Productization Requirements

13e. Release Requirements

13f. Backwards Compatibility Requirements

14. Maintainability and Support Requirements

14a. Maintenance Requirements

14b. Supportability Requirements

14c. Adaptability Requirements

15. Security Requirements

15a. Access Requirements

15b. Integrity Requirements

15c. Privacy Requirements

15d. Audit Requirements

15e. Immunity Requirements

16. Cultural Requirements

16a. Cultural Requirements

17. Compliance Requirements

17a. Legal Compliance Requirements

17b. Standards Compliance Requirements

Project Issues

18. Open Issues

19. Off-the-Shelf Solutions

19a. Ready-Made Products

19b. Reusable Components

19c. Products That Can Be Copied

20. New Problems

20a. Effects on the Current Environment

20b. Effects on the Installed Systems

20c. Potential User Problems

20d. Limitations in the Anticipated Implementation Environment That May Inhibit the New Product

20e. Follow-Up Problems

21. Tasks

21a. Project Planning

21b. Planning of the Development Phases

22. Migration to the New Product

22a. Requirements for Migration to the New Product

22b. Data That Has to Be Modified or Translated for the New Product

23. Risks

24. Costs

25. User Documentation and Training

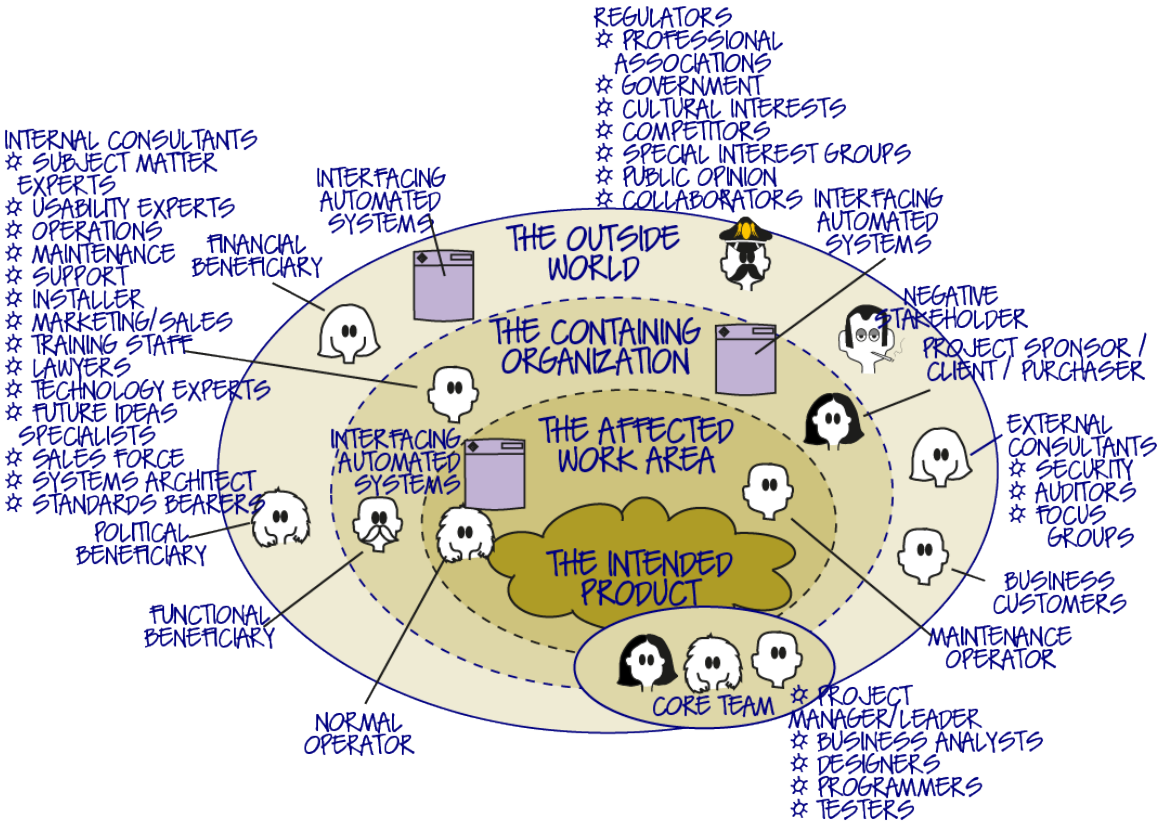
25a. User Documentation Requirements

25b. Training Requirements

26. Waiting Room

27. Ideas for Solutions

Volere Stakeholder Map

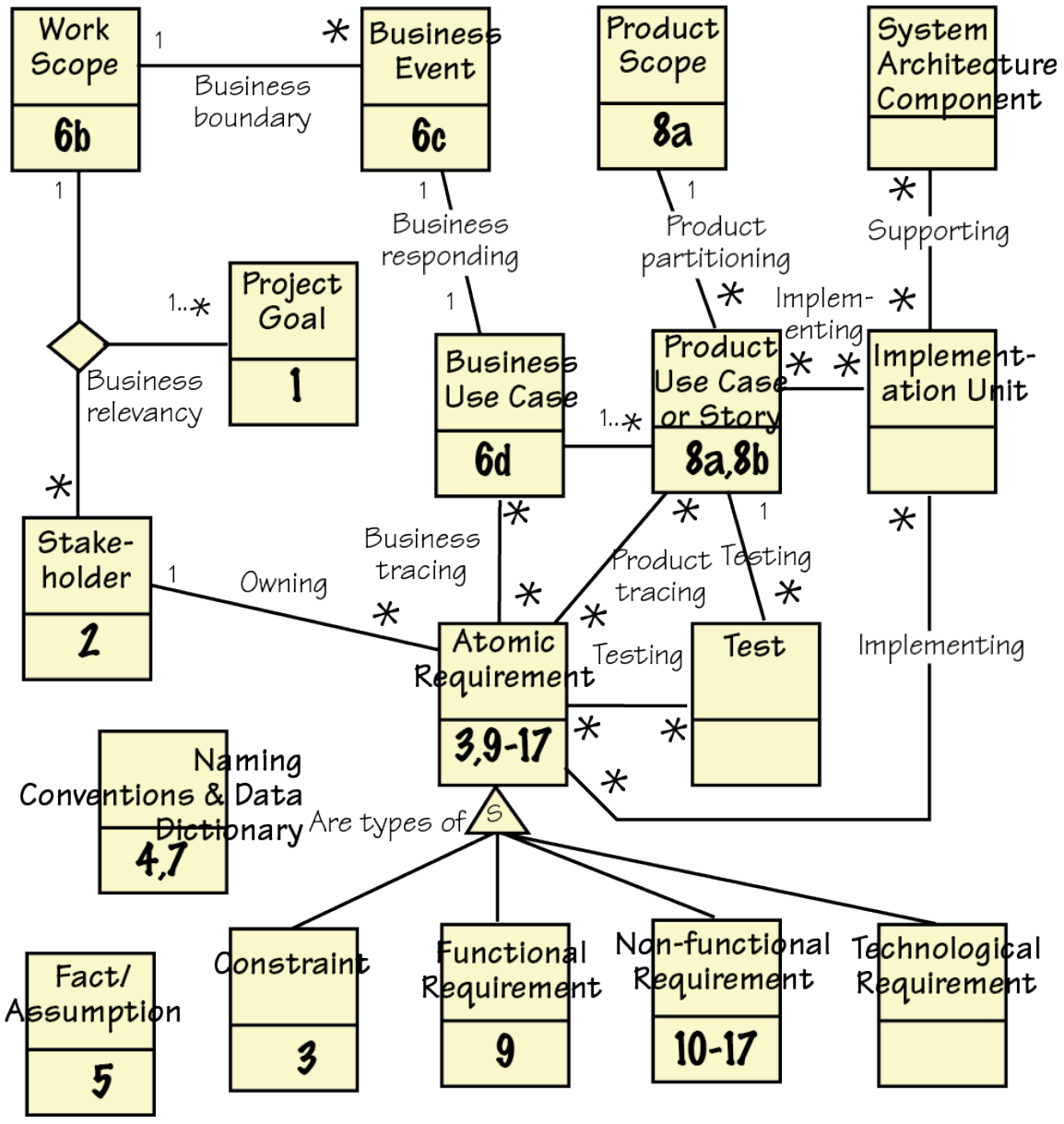


Volere Requirements Knowledge Model

The requirements knowledge model provides a language for collecting and communicating the information that you discover during requirements-related activities. We present it here as a guide to the information you need to consider, and as a tool for communication between the various stakeholders on your project.

The model also serves as a planning tool; you construct it to guide the requirements team as to what information should be gathered. Your own process must define who gathers which information, to what degree of detail and how it will be packaged and reviewed. The model below shows the links between the various classes of knowledge. Typically it is necessary to maintain these links for traceability purposes.

The numbers on the classes provide a cross reference to the relevant section(s) of the Volere template.



The knowledge model identifies the classes of knowledge concerned with requirements and the relationships between them.

The notation used is UML class diagram. If you are not familiar with this notation, it is read as follows:

- The rectangle represents a **class or entity** of knowledge, the name of the class is written in the rectangle
- The line joining rectangles represents a **relationship or association** between two or more classes
- **Multiplicity or cardinality** is represented by 1 (one) and * (many).

Definitions of Requirements Knowledge Classes and Relationships

It is not enough to have just names for classes of knowledge, you also need to have a common agreement of what those names mean. For example, if you have a class on your knowledge model called *Atomic Requirement* then your understanding of what that means should be identical to that of the rest of your team. In other words you need a dictionary to support the model you use. The following are definitions of the classes of knowledge and their associations. First we have listed the classes in alphabetical order and then the relationships in alphabetical order.

Knowledge Classes:

Atomic Requirement

Purpose: A Requirement specifies a business need or want. A requirement has a number of attributes as listed below.

Attributes:

Requirement Number
Requirement Description
Requirement Rationale
Requirement Type
Requirement Fit Criterion
Requirement Source
Customer Satisfaction
Customer Dissatisfaction
Conflicting Requirements
Dependent Requirements
Supporting Material
Version Number

Considerations: Also see the subtypes of requirement, namely Constraint, Functional requirement, Non-functional requirement, Technological requirement.

Suggested Implementation: Sections 9 through 17 of the Template. There are various automated tools available; these allow team access to the requirements.

Business Event

Purpose: A Business Event is a happening outside the work scope that is in effect a demand for some service provided by the work. For example, a motorist passes an electronic tollbooth, a customer orders a book, a doctor asks for the scan of a patient, a pilot lowers the landing gear.

Business events can also happen because of the passage of time. For example, if a customer's bill is not paid in 30 days, then it is time for the work to send a reminder. Or it is two months before an insurance policy is due to expire.

Attributes:

Business Event Name
Business Event Adjacent Systems/Actors
Business Event Summary

Considerations: It is important to recognize the business event. Its nature, the circumstances that exist at the time the event happens, the activity of the adjacent system at the time of the business event are all important indicators of the appropriate response.

Suggested Implementation: This is in section 6 of the Volere Requirements Specification Template. A list of the business events and their associated input and output flows will suffice. It is practical to give each business event a unique identifier.

Business Use Case

Purpose: A Business Use Case (often referred to as a BUC) is the processing done in response to a business event. For example, a policyholder decides to make a claim is a business event. The business use case is all the processing done by the work to approve or deny the claim. Also see Product Use Case.

Attributes:

Business Use Case Name
Business Use Case Description
Business Use Case Input
Business Use Case Outputs
Business Use Case Rationale
Business Use Case Priority
Normal Case Scenario
Exception Case Scenarios

Preconditions
Post or exit conditions

Considerations: Business use cases are self-contained portions of the work, and can be studied independently. For this reason they are an important unit that project leaders can use to structure the analytical work.

Suggested Implementation: Also section 6 of the Volere Requirements Specification Template. An automated tool that allows sharing of the business use case attributes is advisable. BUCs can be represented using any combination of Business process models, sequence diagrams, activity diagrams, scenarios or any other representation that is acceptable to the people involved – providing the BUC is within the boundaries declared for the Business Event.

Constraint

Purpose: A constraint is a type of requirement. It is a constraint on the design of the product, or a constraint on the project itself, such as budget or time restrictions.

Considerations: We treat them as a type of requirement that must be met. However, we highlight them, as it is important that you and your management are aware of them.

Suggested Implementation: Section 3 of the Template. Design constraints should be recorded in the same way as the other requirements. See the knowledge class **Requirement** for the attributes.

Fact/Assumption

Purpose: An assumption states an expectation on which decisions about the project are based. For example, it might be an assumption that another project will be finished first, or that a particular law will not be changed or that a particular supplier will reach a specified level of performance. If an assumption turns out not to be true, then it indicates that there might be far reaching and unknown effects on the project.

A Fact is some knowledge that is relevant to the project and affects its requirements and design. A Fact can also state some specific exclusion from the product and the reason for that exclusion.

Fact/Assumption is a global class and could have an association with any of the other classes in your knowledge model.

Attributes:

Description of the Assumption/Fact
Reference to people and documents for more details

Considerations: Assumptions indicate a risk. For this reason they should be highlighted and all affected parties made aware of the assumption. You could consider installing a mechanism to resolve all assumptions before implementation starts.

Suggested Implementation: Section 5 of the Template; these can be written in free text. They should be regularly circulated to management and the project team.

Functional Requirement

Purpose: A functional requirement is something that the product must do. For example, calculate the fare, analyze the chemical composition, record the change of name, find the new route. Functional requirements are concerned with creating, updating, referencing and deleting the essential subject matter within the context of the study.

Attributes: This is a sub-type of **Atomic Requirement** and inherits its attributes.

Suggested Implementation: Section 9 of the Template. See the Knowledge class **Atomic Requirement** for the attributes.

Implementation Unit

Purpose: The unit for packaging your implementation.

Attributes:

Implementation Unit Name

Considerations: This could be what your customers refer to as a "feature", or if your product is a consumer item then it is possibly called a "function". The choice of implementation unit is driven by a combination of your implementation technology and your implementation process. When you tailor this part of the knowledge model you might find that you replace implementation unit with several classes. The important issue is that you can unambiguously trace your implementation unit back to the relevant requirements.

Naming Conventions & Data Dictionary

Purpose: A dictionary that defines the meaning of terms used within the requirements. This dictionary will be added to throughout the project to include terms that are related to the implementation. This is a global class and could have an association with any of the other classes in your knowledge model. Consistent use of the same terminology – as defined in the dictionary- helps to minimize misunderstandings.

Attributes:

Name of the Term
Definition of the Term

Suggested Implementation: Section 4 of the Template; this should be in the form of a **glossary**. Along with the work context and the product context this provides a good introduction for new team members. In Section 7 of the Template; there is a formal **dictionary** that defines all of the data in the inputs, outputs and attributes within the scope of the work and the scope of the product. The dictionary provides a mechanism for connecting business terminology and implementation terminology.

Non-functional Requirement

Purpose: A Non-functional requirement is a quality that the product must have. For example it must be fast, attractive, secure, customizable, maintainable, portable, etc. Non-functional requirements types are Look and Feel, Usability, Performance and Safety, Operational Environment, Maintainability and Portability, Security, Cultural and Political, Legal. For more about non-functional requirements refer to the Volere requirements template at http://www.volere.co.uk

Attributes: This is a sub-type of **Atomic Requirement** and inherits its attributes.

Considerations: The non-functional properties are important if the user or buyer is to accept the product.

Suggested Implementation: Sections 10 through 17 of the Template. It is vital that you give all non-functional requirements the correct fit criterion.

Product Scope

Purpose: The product scope identifies the boundaries of the product that will be built. The scope is a summary of the boundaries of all the product use cases.

Attributes:

User Names
User Roles
Other Adjacent Systems
Interface descriptions

Suggested Implementation: Section 8 of the Template. This should preferably be a diagram, either a use case diagram or a product scope model supported by a product use case summary table. Some interface descriptions might be supported by prototypes or simulations.

Product Use Case

Purpose: A Product Use Case (PUC) is a functional grouping of requirements that will be implemented by the product. It is that part of the business use case that you decide to build as a product.

Attributes:

Product Use Case Name
Product Use Case Identifier
Product Use Case Description
Product Use Case Users
Product Use Case Inputs
Product Use Case Outputs
Product Use Case Stories
Product Use Case Scenarios
Product Use Case Fit Criterion
Product Use Case Owner
Product Use Case Benefit
Product Use Case Priority

Suggested Implementation: Section 8 of the Template. The product use cases are a good mechanism for communication within the extended project team. PUC's might take the form of models, user stories, scenarios or anything else that suits the people involved. Whatever the type of

representation the details of the PUC should be within the boundaries declared by the PUC inputs and outputs.

Project Goal

Purpose: To understand why the company is making an investment in doing this project.

Attributes:

Project Goal Description
Business Advantage
Measure of Success

Note that there might be several project goals.

Suggested Implementation: Section 1 of the Template. This is the basis for making decisions about scope, relevance and priority. It is the guiding light for the project. Ideally this should be defined as part of the project initiation. The project goals should be unambiguously defined and agreed before putting effort into discovering detailed requirements.

Stakeholder

Purpose: Identifies all the people, roles, organizations who have an interest in the project. This covers the project team, direct users of the product, other indirect beneficiaries of the product, specialists with technical skills needed to build the product, external organizations with rules or laws pertaining to the product, external organizations with specialist knowledge about the product's domain, opponents of the product, producers of competitive products.

Attributes:

Stakeholder Role
Stakeholder Name
Types of Knowledge
Necessary Participation
Appropriate Trawling Techniques
Contact information e.g. email address.

Suggested Implementation: Section 2 of the Template. Use the stakeholder map and stakeholder analysis template to define the attributes for each stakeholder.

Story

Purpose: A Story is a grouping of requirements that reflect a need from the point of view of a particular stakeholder. Stories are used as an alternative to product use cases. If you are working in an agile environment then it is likely that you will be using stories. Stories, like product use cases, provide a functional grouping for detailed atomic requirements – the difference is that stories are at a lower level of granularity.

Attributes:

Role of Stakeholder
Something the Stakeholder needs to be able to do
The reason for wanting this functionality

Suggested Implementation:

As a [Name of Stakeholder]

I can [Something I need to be able to do]

So that [The reason/rationale behind the need]

System Architecture Component

Purpose: A piece of technology, software, hardware or abstract container, that influences, facilitates and /or places constraints on the design.

Technological Requirement

Purpose: A technological requirement exists because of the technology chosen for the implementation. These requirements are there to serve the purposes of the technology, and are not originated by the business.

Attributes: This is a sub type of **Atomic Requirement** and inherits its attributes.

Considerations: The technological requirements should only be considered when you know the technological environment. They can be recorded alongside the business requirements, but it must be clear which is which.

Test

Purpose: The design for test is the result of a tester reviewing a requirement's fit criterion (precise measure) and designing a cost effective test to prove whether or not a solution meets the fit criterion.

Considerations: You might consider having your testing people write the test cases as the requirements are being written. Also consider that the requirement's fit criterion is the basis of the test case.

Work Scope

Purpose: Defines the boundary of the investigation necessary to discover, invent, understand and identify the requirements for the product.

Attributes:

Adjacent Systems
Input Dataflows
Output Dataflows
Work Context Description

Considerations: This should be recorded publicly as our experience is that it is the most widely referenced document. A context model is an effective communication tool for defining the work context.

Suggested Implementation: Section 7 of the Template. The work scope is best illustrated with a context model.

Associations

Business boundary

Purpose: To partition the work context according to the functional reality of the business.

Multiplicity:

For each Business Event there is one Work Context.

For each Work Context there are potentially many Business Events.

Business relevancy

Purpose: To ensure that there are relevant business connections between the scope of the investigation, the project purpose and the stakeholders.**Multiplicity**:

The trinary Relationship is as follows:

For each instance of

one Work Context and

one Stakeholder there are one or more Project Purposes.

For each instance of

one Project Purpose and

one Stakeholder there is one Work Context.

For each instance of

one Project Purpose and

one Work Context there are potentially many Stakeholders.

Business responding

Purpose: To reveal which business use cases are used to respond to the business event.

Multiplicity:

For each Business Event there is usually one, but could be more than one, Business Use Cases.

For each Business Use Case there can only be one triggering Business Event.

Business tracing

Purpose: To keep track of which requirements are generated by which business use cases. Note that this is a many-to-many Relationship because a given requirement might exist in more than one business use case.

Multiplicity:

For each Business Use Case there are potentially many Atomic Requirements.

For each Atomic Requirement there are potentially many Business Use Cases.

Implementing

Purpose: To keep track of which product use cases are implemented in which implementation units.

Multiplicity:

For each Product Use Case there are potentially many Implementation Units.

For each Implementation Unit there are potentially many Product Use Cases.

Owning

Purpose: To keep track of which stakeholders are the source of which requirements. The idea of "ownership" is to identify a person who takes the responsibility for helping to get answers to questions about the requirement.

Multiplicity:

For each Requirement there is one Stakeholder.

For each Stakeholder there are potentially many Requirements.

Product partitioning

Purpose: All the product use cases together form the complete scope of the product. The product scope is partitioned into a number of product use cases.

Multiplicity:

For each Product Use Case there is one Product Scope.

For each Product Scope there are potentially many Product Use Cases.

Product tracing

Purpose: To keep track of which requirements are contained in which product use cases for the purpose of traceability and dealing with change.

Multiplicity:

For each Requirement there are potentially many Product Use Cases.

For each Product Use Case there are potentially many Atomic Requirements.

Supporting

Purpose: To keep track of which systems architecture components support which implementation units for the purpose of tracking tests and assessing impact of change.

Multiplicity:

For each System Architecture Component there are potentially many Implementation Units.

For each Implementation Unit there are potentially many System Architecture Components.

Testing

Purpose: To keep track of which atomic requirements or PUC related groups of atomic requirements are covered by which tests.

Multiplicity:

For each Test there are potentially many Atomic Requirements.

Bibliography

Ackoff, Russell, and Herbert Addison. *Systems Thinking for Curious Managers: With 40 New Management f-Laws*. Triarchy Press, 2010.

Alexander, Ian and Ljerka Beus-Dukic. *Discovering Requirements: How to Specify Products and Services*. Wiley, 2009. §

Alexander, Ian, Neil Maiden, et al. *Scenarios, Stories, Use Cases Through the Systems Development Life-Cycle*. John Wiley, 2004.

Alexander, Ian, and Richard Stevens. *Writing Better Requirements.* Addison-Wesley, 2002.

Beyer, Hugh, and Karen Holtzblatt. *Contextual Design: Defining Customer-Centered Systems*. Morgan Kauffmann, 1998.

Blais, Steven. *Business Analysis – Best Practices for Success*. IIL/Wiley, 2012.

Boehm, Barry, and Richard Turner. *Balancing Agility and Discipline: A Guide for the Perplexed*. Addison-Wesley, 2004.

Booch, Grady, James Rumbaugh, and Ivar Jacobson. *Unified Modeling Language User Guide,* second edition. Addison-Wesley, 2005.

Brooks, Fred. *The Design of Design*. Addison-Wesley, 2010.

Buzan, Tony, and Chris Griffiths. *Mind Maps for Business: Revolutionise Your Business Thinking and Practise*. BBC Active, 2009.

Checkland, Peter, and J. Scholes. *Soft Systems Methodology in Action.* John Wiley & Sons, 1991.

Cockburn, Alastair. *Agile Software Development: The Cooperative Game*. Addison-Wesley, 2006.

Cockburn, Alistair. *Writing Effective Use Cases*. Addison-Wesley, 2001.

Cohn, Mike. *User Stories Applied: For Agile Software Development*. Addison-Wesley, 2004.

Cooper, Alan. *The Inmates Are Running the Asylum: Why High Tech Products Drive Us Crazy and How to Restore the Sanity*. Sams Publishing, 2004.

Cooper, Alan, Robert Reimann, and David Cronin. *About Face 3: The Essentials of Interaction Design,* third edition. Wiley, 2007.

Davis, Alan. *Just Enough Requirements Management*. Dorset House, 2005.

DeMarco, Tom, Peter Hruschka, Tim Lister, Steve McMenamin, James Robertson, and Suzanne Robertson. *Adrenaline Junkies and Template Zombies: Patterns of Project Behaviour.* Dorset House, 2009.

Ferdinandi, Patricia. *A Requirements Pattern: Succeeding in the Internet Economy*. Addison-Wesley, 2001.

Fowler, Martin. *UML Distilled: A Brief Guide to the Standard Object Modeling Language,* third edition. Addison-Wesley, 2003.

Garmus, David, and David Herron. *Function Point Analysis: Measurement Practices for Successful Software Projects*. Addison-Wesley, 2000.

Gause, Donald, and Gerald Weinberg. *Are Your Lights On? How to Figure out What the Problem Really Is.* Dorset House, 1990.

Gause, Donald, and Gerald Weinberg. *Exploring Requirements: Quality Before Design*. Dorset House, 1989.

Gilb, Tom. *Competitive Engineering: A Handbook for Systems Engineering, Requirements Engineering, and Software Engineering Using Planguage*. Butterworth-Heinemann, 2005.

Gottesdiener, Ellen. *Requirements by Collaboration: Workshops for Defining Needs*. Addison-Wesley, 2002.

Highsmith, James. *Adaptive Software Development: A Collaborative Approach to Managing Complex Systems*. Dorset House, 2000.

Holtzblatt, Karen, Jessamyn Burns Wendell, and Shelley Wood. *Rapid Contextual Design: A How-to Guide to Key Techniques for User-Centered Design.* Morgan Kaufmann, 2004.

Hull, Elizabeth, Ken Jackson, and Jeremy Dick. *Requirements Engineering,* second edition. Springer, 2005.

International Institute of Business Analysts. *The Agile Extension to the BABOK Guide.* IIBA, Canada, 2015.

International Institute of Business Analysts. *The Business Analysis Body of Knowledge BABOK Version 3.* IIBA, Canada, 2015.

Jackson, Michael. *Problem Frames: Analyzing and Structuring Software Development Problems.* Addison-Wesley, 2001.

Jones, Capers. *Applied Software Measurement*. McGraw-Hill, 1991.

Kruchten, Philippe. *The Rational Unified Process: An Introduction,* third edition. Addison-Wesley, 2003.

Laplante, Phillip. *Requirements Engineering for Software and Systems*. Auerbach Publications, 2009.

Lauesen, Soren. *Software Requirements: Styles & Techniques*. Addison-Wesley, 2002.

Lawrence-Pfleeger, Shari. *Software Engineering: Theory and Practice,* fourth edition. Prentice Hall, 2009.

Leffingwell, Dean. *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise*. Addison-Wesley, 2011.

Leffingwell, Dean, and Don Widrig. *Managing Software Requirements: A Use Case Approach,* second edition. Addison-Wesley, 2003.

Maiden, Neil, and Suzanne Robertson. *Integrating Creativity into Requirements Processes: Experiences with an Air Traffic Management System*. International Conference on Software Engineering, May 2005.

McMenamin, Steve, and John Palmer. *Essential Systems Analysis*. Yourdon Press, 1984.

Meadows, Donella. *Thinking in Systems: A Primer*. Chelsea Green Publishing, 2008.

Miller, Roxanne. *The Quest for Software Requirements.* MavenMark Books, 2009.

Pfleeger, Charles, and Shari Lawrence Pfleeger. *Security in Computing,* fourth edition. Prentice Hall, 2006.

Pilone, Dan, and Neil Pitman. *UML 2.0 in a Nutshell*. O'Reilly, 2005.

Podeswa, Howard. *The Business Analyst's Handbook*. Course Technology, 2008.

Pohl, Klaus. *Requirements Engineering: Fundamentals, Principles, and Techniques*. Springer, 2010.

Pohl, Klaus and Chris Rupp. *Requirements Engineering Fundamentals: A Study Guide for the Certified Professional for Requirements Engineering*. Rocky Nook, 2015.

Pullen, Penny and James Archer (editors). *Business Analysis and Leadership – Influencing Change*. Kogan Page, 2013.

Robertson, James, and Suzanne Robertson. *Complete Systems Analysis: The Workbook, the Textbook, the Answers.* Dorset House, 1994.

Robertson, Suzanne, and James Robertson. *Mastering the Requirements Process,* second edition. Addison-Wesley, 2006.

Robertson, Suzanne, and James Robertson. *Requirements-Led Project Management: Discovering David's Slingshot.* Addison-Wesley, 2005.

Rumbaugh, James, Ivar Jacobson, and Grady Booch. *Unified Modeling Language Reference Manual,* second edition. Addison-Wesley, 2004.

Seddon, John, *Systems Thinking in the Public Sector*. Triarchy Press, 2010.

Senge, Peter. *The Fifth Discipline: The Art and Practice of the Learning Organization,* revised edition. Crown Business, 2006.

Sommerville, Ian, and Pete Sawyer. *Requirements Engineering: A Good Practice Guide*. John Wiley & Sons, 1998.

Spolsky, Joel. *Joel on Software: And on Diverse and Occasionally Related Matters That Will Prove of Interest to Software Developers, Designers, and Managers, and to Those Who, Whether by Good Fortune or Ill Luck, Work with Them in Some Capacity*. Apress, 2004.

Tockey, Steve. *Return on Software: Maximizing the Return on Your Software Investment*. Addison-Wesley, 2004.

Tufte, Edward. *The Visual Display of Quantitative Information,* second edition. Graphics Press, 2010.

Van Lamsweerde, Axel. *Requirements Engineering.* John Wiley & Sons, 2009.

Wiegers, Karl. *More About Software Requirements: Thorny Issues and Practical Advice*. Microsoft Press, 2006.

Weigers, Karl and Joy Beatty. *Software Requirements,* third edition. Microsoft Press, 2013.

Weinberg, Jerry. *Quality Software Management. Volume 1: Systems Thinking. Volume 2: First-Order Measurement. Volume 3: Congruent Action. Volume 4: Anticipating Change.* Dorset House, 1992-1997.

Wiley, Bill. *Essential System Requirements: A Practical Guide to Event-Driven Methods*. Addison-Wesley, 1999.

Yayici, Emrah. *Business Analyst's Mentor Book : With Best Practice Business Analysis Techniques and Software Requirements Management Tips*. Kindle Edition, 2013.

