Chapter 1

Establishing the business requirements

As you saw in Chapter ??, "The essential software requirement," business requirements represent the top of the requirements chain. They define the vision of the solution and the scope of the project that will implement the solution. The user requirements and functional requirements must align with the context and objectives that the business requirements establish. Requirements that don't help the project achieve its business objectives shouldn't be implemented.

This chapter describes the vision and scope document, a deliverable that contains the project's business requirements.

1.1 Defining business requirements

"Business requirements" refers to a set of information that, in the aggregate, describes a need that leads to one or more projects to deliver a solution and the desired ultimate business outcomes. Business opportunities, business objectives, success metrics, and a vision statement make up the business requirements.

Business requirements issues must be resolved before the functional and nonfunctional requirements can be fully specified. A statement of the project's scope and limitations helps greatly with discussions of proposed features and target releases.

1.1.1 Product vision and project scope

Two core elements of the business requirements are the vision and the scope. The product vision succinctly describes the ultimate product that will achieve the business objectives. This product could serve as the complete solution for the business requirements or as just a portion of the solution.

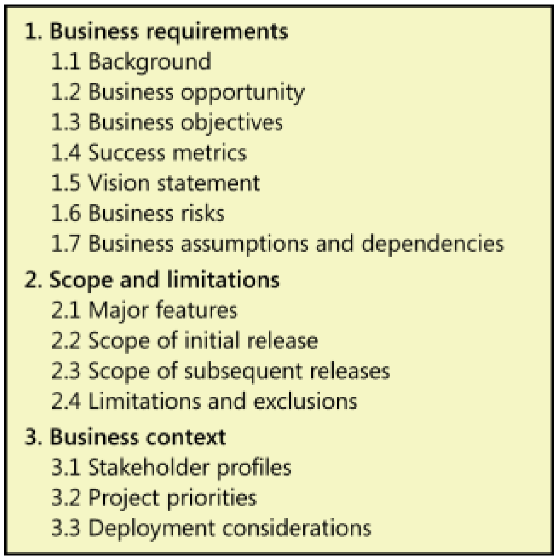
The vision describes what the product is about and what it ultimately could become. It provides the context for making decisions throughout the product's life, and it aligns all stakeholders in a common direction. The project scope identifies what portion of the ultimate product vision the current project or development iteration will address. The statement of scope draws the boundary between what's in and what's out for this project.

1.2 Vision and scope document

The vision and scope document collects the business requirements into a **single deliverable** that sets the stage for the subsequent development work.

A business analyst can work with this individual to articulate the business requirements and write the vision and scope document. Input to the business requirements should come from people who have a clear sense of why they are undertaking the project. These individuals might include the customer or development organization's senior management, a product visionary, a product manager, a subject matter expert, or members of the marketing department.
Figure 1.1 suggests a template for a vision and scope document; the sections that follow describe each of the template headings in more detail. As with any template, adapt this to meet the specific needs of your own projects. If you already have recorded some of this information elsewhere, do not duplicate it in the vision and scope document. Some elements of the vision and scope document might be reusable from project to project, such as business objectives, business risks, and stakeholder profiles.

Figure 1.1: Suggested template for a vision and scope document.



1. Business requirements
- 1.1 Background
- 1.2 Business opportunity
- 1.3 Business objectives
- 1.4 Success metrics
- 1.5 Vision statement
- 1.6 Business risks
- 1.7 Business assumptions and dependencies

2. Scope and limitations
- 2.1 Major features
- 2.2 Scope of initial release
- 2.3 Scope of subsequent releases
- 2.4 Limitations and exclusions

3. Business context
- 3.1 Stakeholder profiles
- 3.2 Project priorities
- 3.3 Deployment considerations

As with any template, adapt this to meet the specific needs of your own projects. If you already have recorded some of this information elsewhere, do not duplicate it in the vision and scope document. Some elements of the vision and scope document might be reusable from project to project, such as business objectives, business risks, and stakeholder profiles.

The vision and scope document only defines the scope at a high level; the scope details are represented by each release baseline that the team defines. Major new projects should have both a complete vision and scope document and an SRS.

1.2.1 Business requirements

The business requirements describe the primary benefits that the new system will provide to its sponsors, buyers, and users. Business requirements directly influence which user requirements to implement and in what sequence.

1.2.1.1 Business opportunity

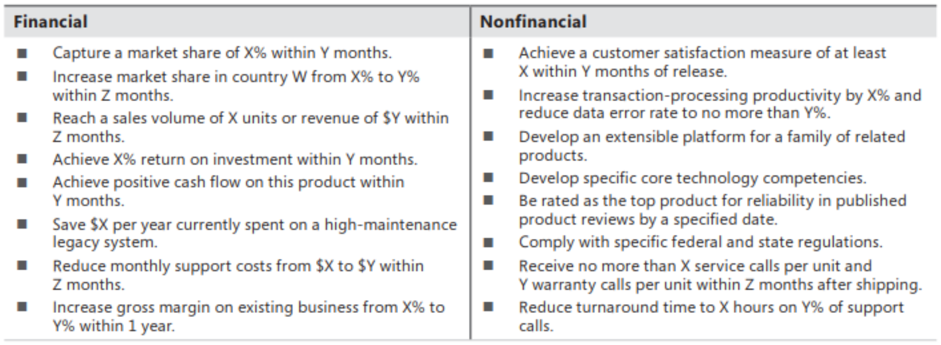
For a corporate information system, describe the business problem that is being solved or the process being improved, as well as the environment in which the system will be used. For a commercial product, describe the business opportunity that exists and the market in which the product will be competing.
Describe the problems that cannot currently be solved without the envisioned

solution. Show how it aligns with market trends, technology evolution, or corporate strategic directions. List any other technologies, processes, or resources required to provide a complete customer solution.

1.2.1.2 Business objectives

Summarize the important business benefits the product will provide in a quantitative and measurable way.
Table 1.1 presents some simplified examples of both financial and nonfinancial business objectives [12].

Table 1.1: Examples of financial and nonfinancial business objectives.



Financial	Nonfinancial
■ Capture a market share of X% within Y months.	■ Achieve a customer satisfaction measure of at least X within Y months of release.
■ Increase market share in country W from X% to Y% within Z months.	■ Increase transaction-processing productivity by X% and reduce data error rate to no more than Y%.
■ Reach a sales volume of X units or revenue of $Y within Z months.	■ Develop an extensible platform for a family of related products.
■ Achieve X% return on investment within Y months.	■ Develop specific core technology competencies.
■ Achieve positive cash flow on this product within Y months.	■ Be rated as the top product for reliability in published product reviews by a specified date.
■ Save $X per year currently spent on a high-maintenance legacy system.	■ Comply with specific federal and state regulations.
■ Reduce monthly support costs from $X to $Y within Z months.	■ Receive no more than X service calls per unit and Y warranty calls per unit within Z months after shipping.
■ Increase gross margin on existing business from X% to Y% within 1 year.	■ Reduce turnaround time to X hours on Y% of support calls.

1.2.1.3 Vision statement

Write a concise vision statement that summarizes the long-term purpose and intent of the product.
The vision statement should reflect a balanced view that will satisfy the expectations of diverse stakeholders. It can be somewhat idealistic but should be grounded in the realities of existing or anticipated markets, enterprise architectures, corporate strategic directions, and resource limitations. The following keyword template works well for crafting a product vision statement [13]:

- For [target customer]
- Who [statement of the need or opportunity]
- The [product name]
- Is [product category]
- That [major capabilities, key benefit, compelling reason to buy or use]

- Unlike [primary competitive alternative, current system, current business process]
- Our product [statement of primary differentiation and advantages of new product]

Here's a sample vision statement for the Chemical Tracking System, with the keywords in boldface:
For *scientists* ***who*** *need to request containers of chemicals,* ***the*** *Chemical Tracking System is an information system* ***that*** *will provide a single point of access to the chemical stockroom and to vendors. The system will store the location of every chemical container within the company, the quantity of material remaining in it, and the complete history of each container's locations and usage. This system will save the company 25 percent on chemical costs in the first year of use by allowing the company to fully exploit chemicals that are already available within the company, dispose of fewer partially used or expired containers, and use a standard chemical purchasing process.* ***Unlike*** *the current manual ordering processes,* ***our product*** *will generate all reports required to comply with federal and state government regulations that require the reporting of chemical usage, storage, and disposal.*

1.2.1.4 Business assumptions and dependencies

An assumption is a statement that is believed to be true in the absence of proof orde finitive knowledge. Business assumptions are specifically related to the business requirements. Incorrect assumptions can potentially keep you from meeting your business objectives. For example, *an executive sponsor might set a business objective that a new website will increase revenue by $100,000 per month. To establish this revenue target, the sponsor made some assumptions, perhaps that the new site will attract 200 additional unique visitors per day and that each visitor will spend an average of $17. If the new site does not attract enough visitors with a high enough average sale per visitor, the project might not achieve its business objective.* If you learn that certain assumptions are wrong, you might have to change scope, adjust the schedule, or launch other projects to achieve the objectives.

Record any assumptions that the stakeholders made when conceiving the project and writing their vision and scope document. Often, one party's assumptions are not shared by others. If you write them down and review them, you can avoid possible confusion and aggravation in the future.

Record any major dependencies the project has on external factors. Examples are *pending industry standards or government regulations, deliverables from other projects, third-party suppliers, or development partners.* Some business assumptions and dependencies might turn into risks that the project manager must monitor regularly. Broken dependencies are a common source of project delays. Note the impact of an assumption not being true, or the impact of a broken dependency, to help stakeholders understand why it is critical.

1.2.2 Scope and limitations

Scope can be represented in numerous ways (see 1.3 "Scope representation techniques" later in this chapter). At the highest level, scope is defined when the customer decides which business objectives to target. At a lower level, scope is defined at the level of features, user stories, use cases, or events and responses to include. Scope ultimately is defined through the set of functional requirements planned for implementation in a specific release or iteration. At each level, the scope must stay within the bounds of the level above it. For example, in-scope user requirements must map to the business objectives, and functional requirements must map to user requirements that are in scope.

1.2.2.1 Major features

List the product's major features or user capabilities, emphasizing those that distinguish it from previous or competing products. Think about how users will use the features, to ensure that the list is complete and that it does not include unnecessary features that sound interesting but don't provide customer value. Give each feature a unique and persistent label to permit tracing it to other system elements. You might include a feature tree diagram, as described later in this chapter.

1.3 Scope representation techniques

Context diagrams, ecosystem maps, feature trees, and event lists are the most common ways to represent scope visually. However, other techniques are also used. Identifying affected business processes also can help define the scope boundary. Use case diagrams can depict the scope boundary between use cases and actors

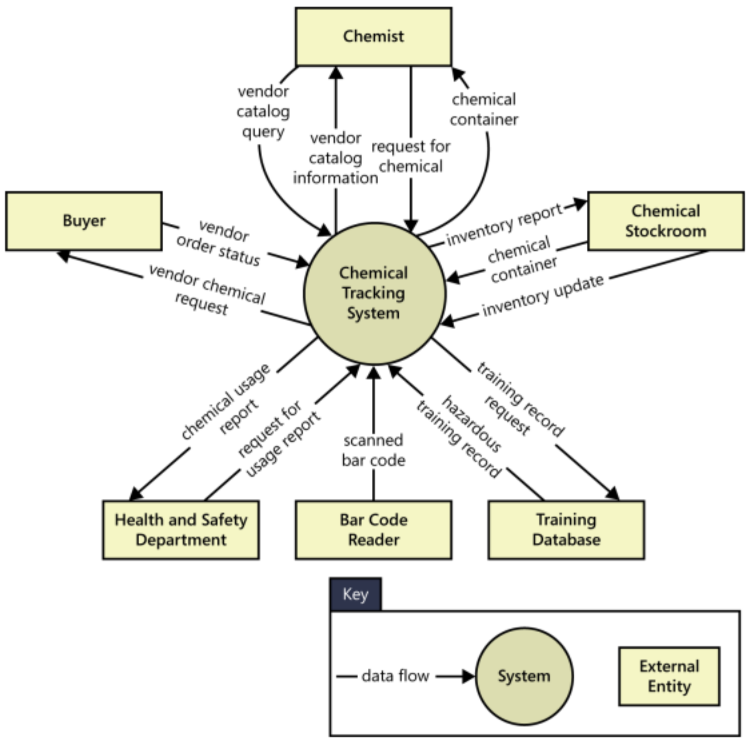
1.3.1 Context diagram

The scope description establishes the boundary and connections between the system you're developing and everything else in the universe. The context diagram visually illustrates this boundary.
It identifies external entities (also called terminators) outside the system that interface to it in some way, as well as data, control, and material flows between the terminators and the system. The context diagram is the top level in a data flow diagram developed according to the principles of structured analysis [14], but it's a useful model for all projects.

Figure 1.2 illustrates a portion of the context diagram for the Chemical Tracking System. The entire system is depicted as a single circle; the context diagram deliberately provides no visibility into the system's internal objects, processes, or data. The "system" inside the circle could encompass any combination of software, hardware, and human components. Therefore, it could include manual operations as part of the entire system. The external entities in the rectangles can represent user classes (Chemist, Buyer), organizations (Health and Safety Department), other systems (Training Database), or hardware devices (Bar Code Reader). The arrows on the diagram represent the flow of data (such as a request for a chemical) or physical items (such as a chemical container) between the system and its external entities.

You might expect to see chemical vendors shown as an external entity in this diagram. After all, the company will route orders to vendors for fulfillment, the vendors will send chemical containers and invoices to Contoso Pharmaceuticals, and Contoso's purchasing department will pay the vendors. However, those processes take place outside the scope of the Chemical Tracking System, as part of the operations of the purchasing and receiving departments. Their absence from the context diagram makes it clear that this system is not directly involved in placing orders with the vendors, receiving the products, or paying the bills.

Figure 1.2: Partial context diagram for the Chemical Tracking System.

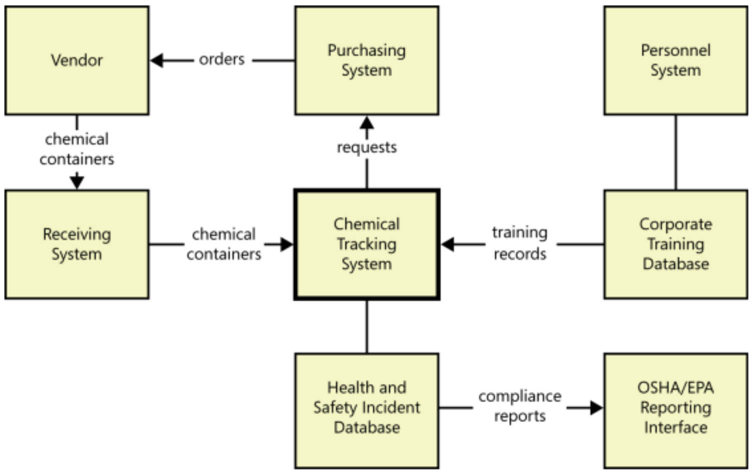


1.3.2 Ecosystem map

An ecosystem map shows all of the systems related to the system of interest that interact with one another and the nature of those interactions [16]. An ecosystem map represents scope by showing all the systems that interconnect and that therefore might need to be modified to accommodate your new system. Ecosystem maps differ from context diagrams in that they show other systems that have a relationship with the system you're working on, including those without direct interfaces. You can identify the affected systems by determining which ones consume data from your system. When you reach the point that your project does not affect any additional data, you've identified the scope boundary of systems that participate in the solution.

Figure 1.3 is a partial ecosystem map for the Chemical Tracking System. The systems are all shown in boxes (such as the Purchasing System or Receiving System). In this example, the primary system we are working on is shown in a bold box (Chemical Tracking System), but if all systems have equal status in your solution, you can use the same box style for all of them. The lines show interfaces between systems (for instance, the Purchasing System interfaces to the Chemical Tracking System).
Lines with arrows and labels show that major pieces of data are flowing from one system to another (for instance, "training records" are passed from the Corporate Training Database to the Chemical Tracking System). Some of these same flows can also appear on the context diagram.

Figure 1.3: Partial ecosystem map for the Chemical Tracking System.



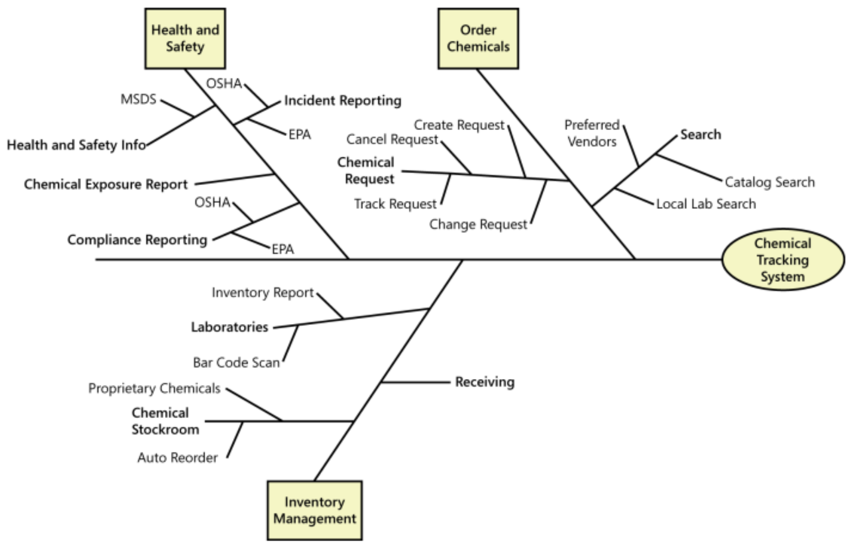
The ecosystem map in 1.3 shows that the Chemical Tracking System does not directly connect to the OSHA/EPA Reporting Interface. Nonetheless, you need to consider whether any requirements in the Chemical Tracking System arise because of the data that flows from it, through the Health and Safety Incident Database, and to that reporting interface.

1.3.3 Feature tree

A feature tree is a visual depiction of the product's features organized in logical groups, hierarchically subdividing each feature into further levels of detail [16]. The feature tree provides a concise view of all of the features planned for a project, making it an ideal model to show to executives who want a quick glance at the project scope. A feature tree can show up to three levels of features, commonly called level 1 (L1), level 2 (L2), and level 3 (L3). L2 features are subfeatures of L1 features, and L3 features are subfeatures of L2 features.

Figure 1.4 shows a partial feature tree for the Chemical Tracking System. The main branch of the tree in the middle represents the product being implemented. Each feature has its own line or "branch" coming off that central main branch. The gray boxes represent the L1 features, such as Order Chemicals and Inventory Management. The lines coming off an L1 branch are L2 features: Search and Chemical Request are subfeatures of Order Chemicals. The branches off an L2 branch are the L3 features: Local Lab Search is a subfeature of Search.

Figure 1.4: Partial feature tree for the Chemical Tracking System.



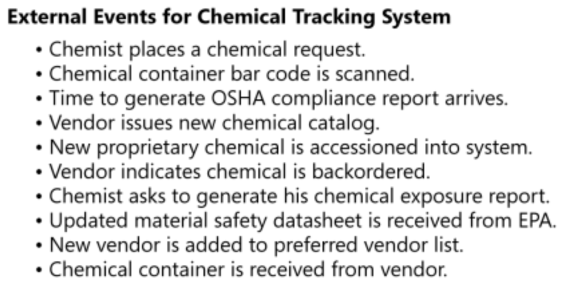
When planning a release or an iteration, you can define its scope by selecting a specific set of features and subfeatures to be implemented ([17]; [11]). You could implement a feature in its entirety in a specific release, or you could implement only a portion of it by choosing just certain L2 and L3 subfeatures. Future releases could enrich these rudimentary implementations by adding more L2 and L3 subfeatures until each feature is fully implemented in the final product. So the scope of a particular release consists of a defined set of L1, L2, and/or L3 features chosen from the feature tree. You can mark up a feature tree diagram to illustrate these feature allocations across releases by using colors or font variations. Alternatively, you can create a feature roadmap table that lists the subfeatures planned for each release [11].

1.3.4 Event list

An event list identifies external events that could trigger behavior in the system. The event list depicts the scope boundary for the system by naming possible business events triggered by users, time-triggered (temporal) events, or signal events received from external components, such as hardware devices. The event list only names the events; the functional requirements that describe how the system responds to the events would be detailed in the SRS by using event-response tables.

Figure 1.5 is a partial event list for the Chemical Tracking System. Each item in the list states what triggers the event ("Chemist" does something or the "Time to" do something arrives), as well as identifying the event action. An event list is a useful scoping tool because you can allocate certain events to be implemented in specific product releases or development iterations.

Figure 1.5: Partial event list for the Chemical Tracking System.



External Events for Chemical Tracking System

- Chemist places a chemical request.
- Chemical container bar code is scanned.
- Time to generate OSHA compliance report arrives.
- Vendor issues new chemical catalog.
- New proprietary chemical is accessioned into system.
- Vendor indicates chemical is backordered.
- Chemist asks to generate his chemical exposure report.
- Updated material safety datasheet is received from EPA.
- New vendor is added to preferred vendor list.
- Chemical container is received from vendor.

Notice how the event list complements the context diagram and ecosystem map. The context diagram and ecosystem map collectively describe the external actors and systems involved, whereas the event list identifies what those actors and systems might do to trigger behavior in the system being specified. You can check the event list against the context diagram and ecosystem map for correctness and completeness, as follows:

- Consider whether each external entity on the context diagram is the source of any events: “Do any actions by Chemists trigger behavior in the Chemical Tracking System?”

- Consider whether any systems in the ecosystem map lead to events for your system.
- For each event, consider whether you have corresponding external entities in the context diagram or systems in the ecosystem map: “If a chemical container can be received from a vendor, does Vendor appear in the context diagram and/or ecosystem map?”

If you find a disconnect, consider whether the model is missing an element. In this case, Vendor did not appear on the context diagram because the Chemical Tracking System doesn’t interface directly to vendors. However, Vendor is included in the ecosystem map.

Bibliography

[1] Bass, Len, Paul Clements, and Rick Kazman. 2013. Software Architecture in Practice, 3nd ed. Reading, MA: Addison-Wesley.

[2] Davis, Alan M. 1995. 201 Principles of Software Development. New York: McGraw-Hill.

[3] Sommerville, Ian, and Pete Sawyer. 1997. Requirements Engineering: A Good Practice Guide. Chichester, England: John Wiley & Sons Ltd.

[4] Box, George E. P., and Norman R. Draper. 1987. Empirical Model-Building and Response Surfaces. New York: John Wiley & Sons, Inc.

[5] Brown, Norm. 1996. "Industrial-Strength Management Strategies." IEEE Software 13(4):94-103.

[6] Kulak, Daryl, and Eamonn Guiney. 2004. Use Cases: Requirements in Context, 2nd ed. Boston: Addison-Wesley.

[7] Cohn, Mike. 2004. User Stories Applied: For Agile Software Development. Boston: Addison-Wesley.

[8] ISO/IEC/IEEE. 2011. "ISO/IEC/IEEE 29148:2011(E), Systems and software engineering—Life cycle processes—Requirements engineering." Geneva, Switzerland: International Organization for Standardization.

[9] Abran, Alain, James W. Moore, Pierre Bourque, and Robert Dupuis, eds. 2004. Guide to the Software Engineering Body of Knowledge, 2004 Version. Los Alamitos, CA: IEEE Computer Society Press.

[10] Beatty, Joy, and Anthony Chen. 2012. Visual Models for Software Requirements. Redmond, WA: Microsoft Press.

[11] Wiegers, Karl E. 2006. More About Software Requirements: Thorny Issues and Practical Advice. Redmond, WA: Microsoft Press.

[12] Wiegers, Karl E. 2007. Practical Project Initiation: A Handbook with Tools. Redmond, WA: Microsoft Press.

[13] Moore, Geoffrey A. 2002. Crossing the Chasm: Marketing and Selling High-Tech Products to Mainstream Customers. New York: HarperBusiness.

[14] Robertson, James, and Suzanne Robertson. 1994. Complete Systems Analysis: The Workbook, the Textbook, the Answers. New York: Dorset House Publishing.

[15] Robertson, Suzanne, and James Robertson. 2013. Mastering the Requirements Process: Getting Requirements Right, 3rd ed. Upper Saddle River, NJ: Addison-Wesley.

[16] Beatty, Joy, and Anthony Chen. 2012. Visual Models for Software Requirements. Redmond, WA: Microsoft Press.

[17] Nejmeh, Brian A., and Ian Thomas. 2002. "Business-Driven Product Planning Using Feature Vectors and Increments." IEEE Software 19(6):34-42.

[18] Booch, Grady, James Rumbaugh, and Ivar Jacobson. 1999. The Unified Modeling Language User Guide. Reading, MA: Addison-Wesley.

[19] Podeswa, Howard. 2009. The Business AnalystŠs Handbook. Boston: Course Technology.

[20] Armour, Frank, and Granville Miller. 2001. Advanced Use Case Modeling: Software Systems. Boston: Addison-Wesley.

[21] Business Rules Group. 2012. http://www.businessrulesgroup.org.

[22] von Halle, Barbara. 2002. Business Rules Applied: Building Better Systems Using the Business Rules Approach. New York: John Wiley & Sons, Inc.

[23] Ross, Ronald G. 1997. The Business Rule Book: Classifying, Defining, and Modeling Rules, Version 4.0, 2nd ed. Houston: Business Rule Solutions, LLC.

[24] Ross, Ronald G., and Gladys S. W. Lam. 2011. Building Business Solutions: Business Analysis with Business Rules. Houston: Business Rule Solutions, LLC.

[25] Ross, Ronald G. 2001. "The Business Rules Classification Scheme." DataToKnowledge Newsletter 29(5).

[26] Morgan, Tony. 2002. Business Rules and Information Systems: Aligning IT with Business Goals. Boston: Addison-Wesley.

[27] von Halle, Barbara, and Larry Goldberg. 2010. The Decision Model: A Business Logic Framework Linking Business and Technology. Boca Raton, FL: Auerbach Publications.

[28] Boyer, Jérôme, and Hafedh Mili. 2011. Agile Business Rule Development: Process, Architecture, and JRules Examples. Heidelberg, Germany: Springer.

[29] Gilb, Tom. 1988. Principles of Software Engineering Management. Harlow, England: Addison-Wesley.

[30] Ambler, Scott. 2005. The Elements of UML 2.0 Style. New York: Cambridge University Press.

[31] Withall, Stephen. 2007. Software Requirement Patterns. Redmond, WA: Microsoft Press.