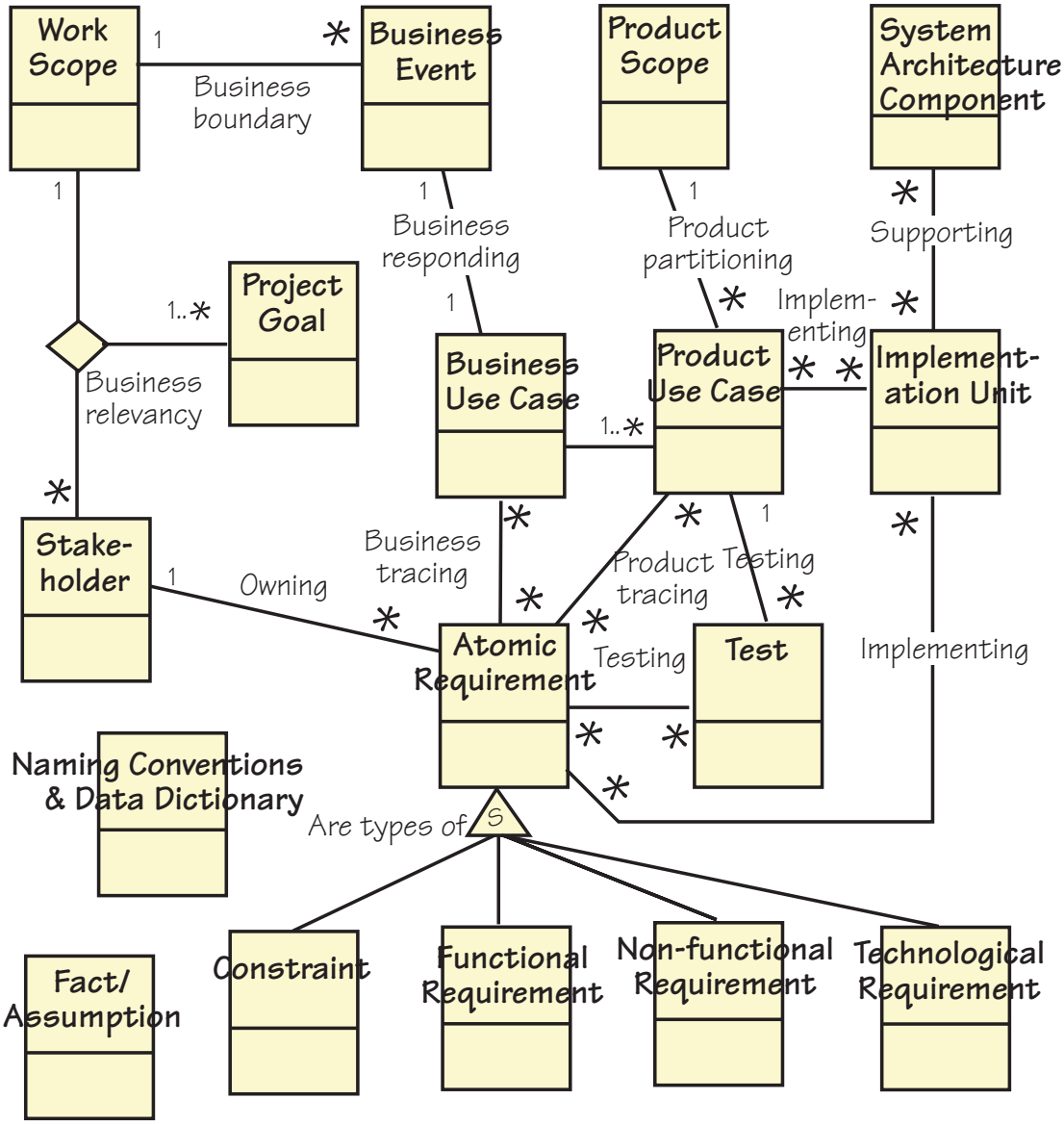
Requirements Knowledge Model

This model provides a language for communicating the knowledge 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 can also serve as the specification for which requirements knowledge you plan to discover and trace. Your own process must define who gathers which information, and how it will be packaged and reviewed.



The knowledge model identifies the classes of knowledge concerned with requirements and the relationships between them. The model uses UML class modelling notation: a rectangle represents a class of knowledge, lines

between classes of knowledge represent relationships, cardinality is represented as 1 (one) and * (many) and can be read in both directions. E.G. one instance of *Work Scope* has a *Business Boundary* relationship with many instances of *Business Event* and one instance of *Business Event* has a *Business Boundary* relationship with one instance of *Work Scope*. The classes and relationships are defined in detail in the knowledge model's data dictionary.

Note that each of the classes on the knowledge model corresponds to one or more of the sections in the Volere requirements template available at http://www.volere.co.uk

Data Dictionary for the Requirements Knowledge Model

The following is a definition of the classes and relationships on the above knowledge model. We have listed the relationships after the classes.

Knowledge class: **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
Requirement Priority
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 though 17 of the Template. There are various automated tools available; these allow team access to the requirements.

Knowledge class: **Business Event**

Purpose:

A Business Event is some 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

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.

Knowledge class: **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.

Knowledge class: **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.

Knowledge class: **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 Relationship 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.

Knowledge class: **Functional Requirement**

Purpose:

A functional requirement is something that the product must do. For example calculate the fare, analyse 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.

Knowledge class: **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.

Knowledge class: **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 Relationship with any of the other classes in your knowledge model. Consistent use of the same terminology – as defined in the dictionary- helps to minimise 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.

Knowledge class: **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 though 17 of the Template. It is vital that you give all non-functional requirements the correct fit criterion.

Knowledge class: **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 context model. Some interface descriptions might be supported by prototypes or simulations.

Knowledge class: **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.

Knowledge class: **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.

Knowledge class: **Stakeholder**

Purpose:

Identifies all the people, roles, organisations 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 organisations with rules or laws pertaining to the product, external organisations 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 eg. email address.

Suggested Implementation:

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

Knowledge class: **System Architecture Component**

Purpose:

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

Knowledge class: **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.

Knowledge class: **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.

Knowledge class: **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. This is best illustrated with a context model or a use case model.

Relationship: **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.

Relationship: **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.

Relationship: **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.

Relationship: **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.

Relationship: **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.

Relationship: **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.

Relationship: **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.

Relationship: **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.

Relationship: **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.

Relationship: **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.