Requirements Diagram Cheat Sheet

A set of requirements diagrams can be thought of as the visual equivalent of a requirements document, and in fact, requirements diagrams, when created according to a set of rules, can be exported to a requirement specification. You can see the key features of a requirements diagram in the No Magic material shown below.

Requirements Dependencies

Trace:
A 'Trace' relationship is a dependency that provides a general purpose relationship between a requirement and any other model elements.

Satisfy:
A 'Satisfy' relationship is a dependency between a requirement and a model element that fulfills that requirement. As with other dependencies, the arrow direction points from the satisfying (client) model element to the (supplier) requirement that is satisfied.

Copy:
A 'Copy' relationship is a dependency between a supplier requirement (master) and a client requirement (slave), specifying that the client requirement text is a read-only copy of the supplier requirement text.

Verify:
A 'Verify' relationship is a dependency between a requirement and a test case or a model element that can determine whether the system fulfills the requirement. Other dependencies, the arrow direction points from the (client) test case to the (supplier) requirement.

Derive:
A 'Derive' relationship is a dependency between two requirements (a derived requirement and a source requirement), where the derived requirement is generated or inferred from the source requirement.

Refine:
A 'Refine' relationship is a dependency intended to describe how a model element or a set of elements are used to further refine a requirement. Alternatively, it can be used to show how a text-based requirement refines a model element.

Possible relationships available for Requirement diagrams are containments, deriveReqt and requirement dependencies ('Copy', 'Refine', 'Satisfy', 'Trace', and 'Verify'). The call out notation can also be used to reflect the relationships of other models. Requirements can also be shown on other diagrams to illustrate their relationships to other modeling elements.

Requirement:
A Requirement specifies a capability or a condition that must (or should) be satisfied. Requirements are used to establish a contract between the customer (or other stakeholders) and those responsible for designing and implementing the system. A requirement can also appear on other diagrams to show its relationship to other modeling elements.

Extended Requirement:
An Extended Requirement adds some properties to the requirement element. These properties are important for requirement management. Specific projects should add their own properties.

Functional Requirement:
A Functional Requirement is a requirement that specifies a behavior that a system or part of a system must perform.

Interface Requirement:
An Interface Requirement is a requirement that specifies the ports for connecting systems and parts of a system. Optionally, it may include the items that flow across the connector and/or the Interface constraints.

Performance Requirement:
A Performance Requirement refers to a requirement that quantitatively measures the extent to which a system or a system part satisfy a required capability or condition.

Physical Requirement:
A Physical Requirement specifies the physical characteristics and/or physical constraints of a system, or a system part

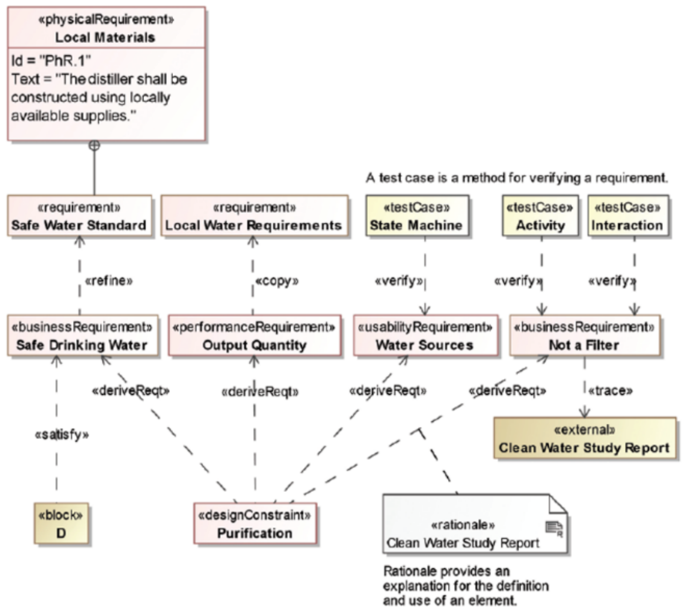
Design Constraint:
A Design Constraint is a requirement that specifies a constraint on the implementation of a system or part of it.

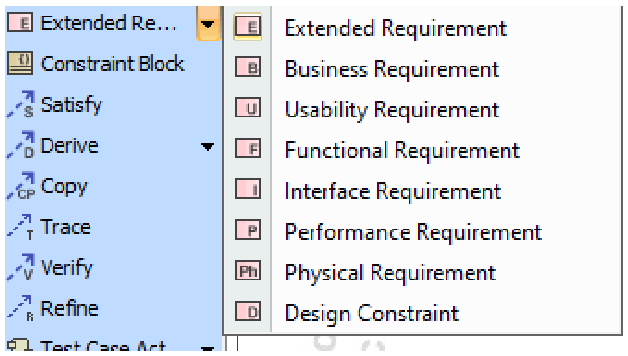
Business Requirement:
A Business Requirement is a requirement that specifies characteristics of the business process that must be satisfied by the system.

Usability Requirement:
A Usability Requirement specifies the fitness for use of a system for its users and other actors.

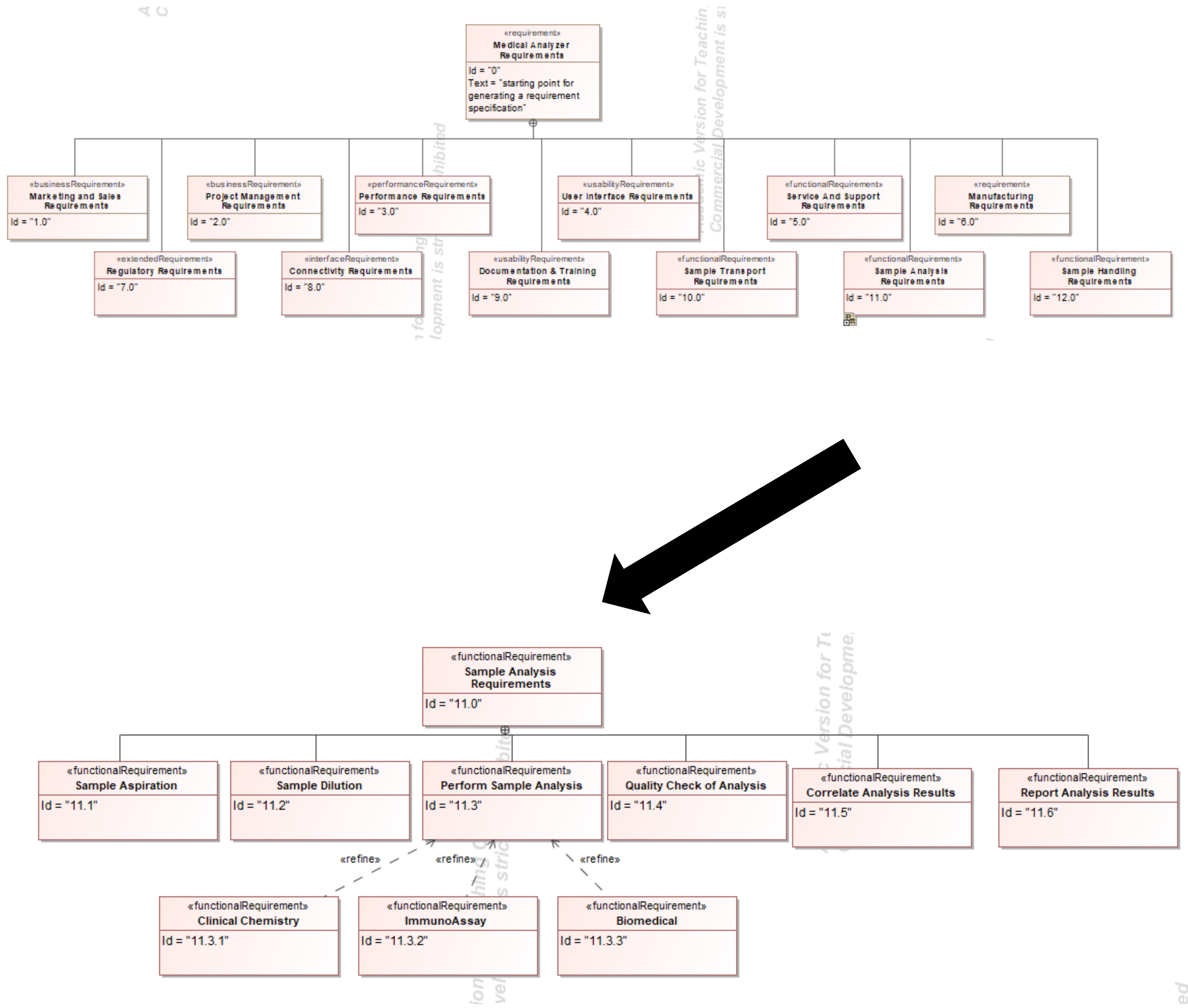
Containment:
This relationship is used to decompose a requirement into its constituent requirements.

Allocation:
an allocation relationship to allocate one model element to another.


«physicalRequirement» Local Materials
Id = "PhR.1"
Text = "The distiller shall be constructed using locally available supplies."
A test case is a method for verifying a requirement.
«requirement» Safe Water Standard
«requirement» Local Water Requirements
«testCase» State Machine
«testCase» Activity
«testCase» Interaction
«refine»
«copy»
«verify»
«verify»
«verify»
«businessRequirement» Safe Drinking Water
«performanceRequirement» Output Quantity
«usabilityRequirement» Water Sources
«businessRequirement» Not a Filter
«deriveReqt»
«deriveReqt»
«deriveReqt»
«deriveReqt»
«trace»
«satisfy»
«external» Clean Water Study Report
«block» D
«designConstraint» Purification
«rationale» Clean Water Study Report
Rationale provides an explanation for the definition and use of an element.



As can be seen on the left, MD comes with a set of stereotyped requirements ready to use. However, users can always add their own stereotypes, e.g. "Safety Requirement" or "User interface Requirement"


«requirement» Medical Analyzer Requirements
Id = "0"
Text = "starting point for generating a requirement specification"
«businessRequirement» Marketing and Sales Requirements
Id = "1.0"
«businessRequirement» Project Management Requirements
Id = "2.0"
«performanceRequirement» Performance Requirements
Id = "3.0"
«usabilityRequirement» User Interface Requirements
Id = "4.0"
«functionalRequirement» Service And Support Requirements
Id = "5.0"
«requirement» Manufacturing Requirements
Id = "6.0"
«extendedRequirement» Regulatory Requirements
Id = "7.0"
«interfaceRequirement» Connectivity Requirements
Id = "8.0"
«usabilityRequirement» Documentation & Training Requirements
Id = "9.0"
«functionalRequirement» Sample Transport Requirements
Id = "10.0"
«functionalRequirement» Sample Analysis Requirements
Id = "11.0"
«functionalRequirement» Sample Handling Requirements
Id = "12.0"
«functionalRequirement» Sample Analysis Requirements
Id = "11.0"
«functionalRequirement» Sample Aspiration
Id = "11.1"
«functionalRequirement» Sample Dilution
Id = "11.2"
«functionalRequirement» Perform Sample Analysis
Id = "11.3"
«functionalRequirement» Quality Check of Analysis
Id = "11.4"
«functionalRequirement» Correlate Analysis Results
Id = "11.5"
«functionalRequirement» Report Analysis Results
Id = "11.6"
«refine»
«refine»
«refine»
«functionalRequirement» Clinical Chemistry
Id = "11.3.1"
«functionalRequirement» ImmunoAssay
Id = "11.3.2"
«functionalRequirement» Biomedical
Id = "11.3.3"

Note that each symbol can be exploded to one or more lower level diagrams. This keeps the number of symbols in each diagram manageable. On most real projects the decomposition continues until every lowest level requirement is testable. Note also that the high-level requirements can correlate with the use cases.