The analysis requirements models

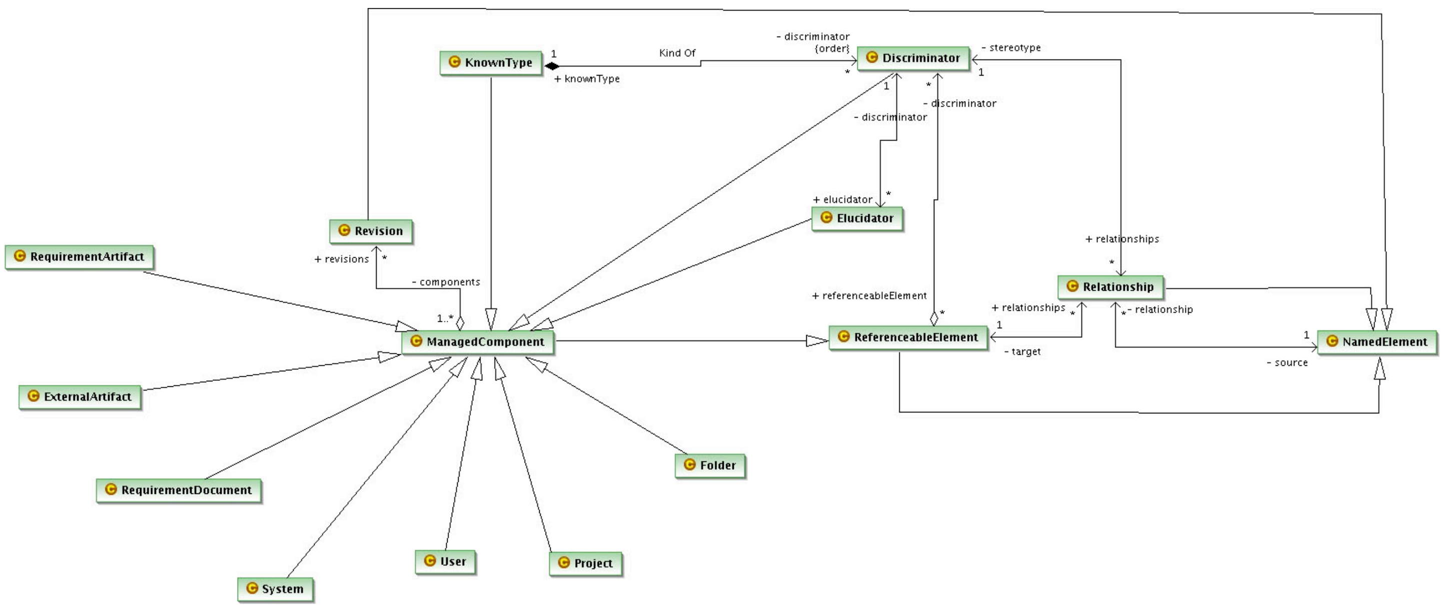
Introduction

This document describes the new requirements model at the initial analytical level. This model follows the review of the originally proposed model and adopts many of the suggestions provided by the review process. The model is currently defined only at a high level and will iteratively refine over the course of the next few weeks.

The high level view of the model

At this stage the model simply reflects the main elements and their mutual relationships.

The figure below shows major players and their most fundamental associations.



These elements are described in the rest of this section.

NamedElement

At the highest level of abstraction we have *NamedElement* (to the right of the diagram above), which represents the most generic element in the model. All model elements are minimally characterised by a name.

ReferenceableElement

The NamedElement is specialised by the *ReferenceableElement*, which, as its name implies, is an element that may be referenced by any other elements in the model.

ManagedComponent

We then have the *ManagedComponent* element, which is unarguably at the core of the model. A ManagedComponent is the smallest ReferenceableElement that is managed directly by the framework. As such, ORMF guarantees to provide all basic CRUD operations on this element, as well as the direct persistification into the repository of the system. This is to be contrasted with other NamedElements, such as a Relationship or an internal constitutive element of a ManagedComponent, which are only created, deleted or persisted as a result of the creation, deletion of persistification of the ManagedComponents to which they contribute.

Furthermore a ManagedComponent is the only element in the model which has the concept of, and an association to, one or more Revision.

ManagedComponents are associated to one or more User playing a specific Role. More details on this relationships will be provided in section Users and Roles below.

Given its central position in the model, ManagedComponent is the abstract parent of a large part of the elements that comprise an ORMF based system.

System

System is a ManagedComponent that represents the entire system that is being created and managed with ORMF. From a containment perspective, the System is the root container of all elements in a requirements model. See section System and related elements below for details on the associations of System with the other elements.

User

User is a ManagedComponent that encapsulates the generic user of an ORMF based system. All issues of authentication and authorisation that are typically connected to a user are handled by ORMF. A User has one or more Roles to play in association with any ManagedComponent. More details on this relationships will be provided in section Users and Roles below. User elements are contained within the System element. See section System and related elements below for details on the associations of User with the other elements.

Project

Project is a ManagedComponent that contains all ManagedComponent elements created to describe the requirements model of a specific software project. Project elements are contained in the System element and may in turn contain any number of Folder elements. See section System and related elements below for details on the associations of Project with the other elements.

Folder

Folder is a ManagedComponent which acts as a container of other Folder elements or ManagedComponents. If not contained by another Folder element, a Folder sits within a Project. See section System and related elements below for details on the associations of Folder with the other elements.

RequirementArtifact

RequirementArtifact is a ManagedComponent which is the common parent to all elements that represent some form of requirement. RequirementArtifact has two specialisations, which will be described in more detail in section Requirements elements below. Section Requirements elements describes also how we see customised RequirementsArtifact elements being added to the system by any adopting organisation.

RequirementDocument

A *RequirementDocument* is a ManagedComponent which represents a textual artefact that acts as a “wrapper” around specific sub sets of ManagedComponents. A good example of this type of element is the Software Requirements Specification document, which is typically a high level survey of all functional, non functional and use case related requirements for a software project or for a specific component of the project. The analyst typically organises the survey in a particular way, that needs to be reproducible through the natural evolution of the requirements and that needs to contain sections of descriptive text that acts as a cohesive agent for all requirement artefacts proper. Examples of such sections are introduction, overview of the problem domain etc. This text is what the model refers to as Augmentation

Text, which is a separate model element, not shown in the diagram above; any Requirement Document makes use of an arbitrary number of Augmentation Text elements.

ExternalArtifact

ExternalArtifact is a ManagedComponent which embodies any external resource (such as an external document, an image or a diagram) that is referred to by (or included in) one or more of the ORMF ManagedComponents. Once the User makes ORMF aware of an ExternalArtifact, it ensures that the artifact exists and can be found when needed.

KnownType and Discriminator

These elements are both ManagedComponent elements. Together they form the basis of the ubiquitous pattern that facilitates extensions of the ORMF framework. Please see section KnownType/Discriminator pattern for a description of these two elements and of the related pattern.

Elucidator

An *Elucidator* is a ManagedComponent that, although not a requirement artefact in its own right, acts as a means to clarify or enrich any NamedElement.

Elucidators are elements that may be utilised across a number of different NamedElements and that permeate the entire requirements model. Examples of Requirements Elucidators are Issues, Glossary Terms and Notes. The distinguishing features of Requirements Elucidator elements are a) the fact that each Elucidator may be referred to by multiple NamedElements (f.i. a Note may be related to, or elucidate, any number of NamedElements) and b) the fact that it is possible to compile lists of all Elucidators of a given type defined across the entire requirements model through generic reporting tools.

The type of Elucidator is defined via the KnownType/Discriminator pattern with KnownType = Elucidator, which is described generically in section KnownType/Discriminator pattern. This ensure easy extensibility of the Elucidator element.

A diagrammatic representation of these two elements and their associations may be found in the top diagram.

Relationship

Relationship is a NamedElement. It represents the association between any NamedElement and any ReferenceableElement, where the NamedElement is the source of the association and the ReferenceableElement is the target.

NamedElement and ReferenceableElement objects may have any number of Relationships defined.

Relationship also carries the concept of an association’s stereotype through its usage of a specific Discriminator object which defines the desired value for a KnownType = Stereotype. The KnownType/Discriminator pattern is described generically in section KnownType/Discriminator pattern. This ensures easy extensibility of stereotypes.

Revision

Revision is a NamedElement which contains all the required information related to revision management and control of any ManagedComponent. A ManagedComponent may be associated with any number of revisions.

KnownType/Discriminator pattern

KnownType is a ManagedComponent element which identifies a type's name, which is indicative of the particular usage being made of the pattern. Examples of KnownType names are "Stereotype", "Elucidator", "Requirement", "Narrative", "Role" and "Priority".

Discriminator is a ManagedComponent element which identifies a possible value for a specific KnownType. When a particular KnownType element may accept more than one value, there will be a Discriminator object for each required value. For example, for the KnownType "Role", there will be, out of the box, at least four Discriminator objects, with values of "Stakeholder", "Owner", "Administrator" and "Project Administrator" respectively. If an adopter requires to identify a new role to be associated with a User, the only things that is needed is to contribute a new Discriminator for KnownType = "Role", with the value of the Discriminator being the desired value. All of the semantics that is necessary to handle the concept of a role is automatically provided by ORMF and therefore ORMF's core does not need to be modified.

The same mechanism can be utilised to extend the system to accept a customised requirement, provided it fits into either the Requirement or the Narrative categories.

Besides adding Discriminators with custom values to existing KnownType elements, adopters may also create their own KnownType and associate one or more Discriminator elements to them. This in itself will be sufficient in most customisation circumstances, as the Discriminator element offers all the required behaviour to accommodate for searching, reporting and filtering based upon specific Discriminator values. We see this as a very powerful tagging mechanism, that will enable adopters to provide specialised visual views into a requirements model or specialised reports or documentation.

Clear examples of popular attributes that can easily be handled simply by the KnownType/Discriminator elements are various management related attributes, such as a requirement's status, priority or complexity. So, for example, a KnownType with name "Priority" may be created, along with three Discriminator objects with values "low", "medium" and "high" respectively. Any ManagedComponent may then simply be tagged with the Priority KnownType and its suitable corresponding Discriminator. All desired selecting, filtering and reporting based upon the Priority KnownType will then be immediately available and easy to activate.

The management related KnownType mentioned above will be included in the framework, but many other may be envisaged based upon a development team's habits or a Project Manager's categorisation preferences.

In those cases in which a more specialised and unique behaviour than the one offered by the plain Discriminator is required, custom elements embodying this behaviour can be created. Examples of such elements in the ORMF model are the Role element and the Relationship element.

Users and Roles

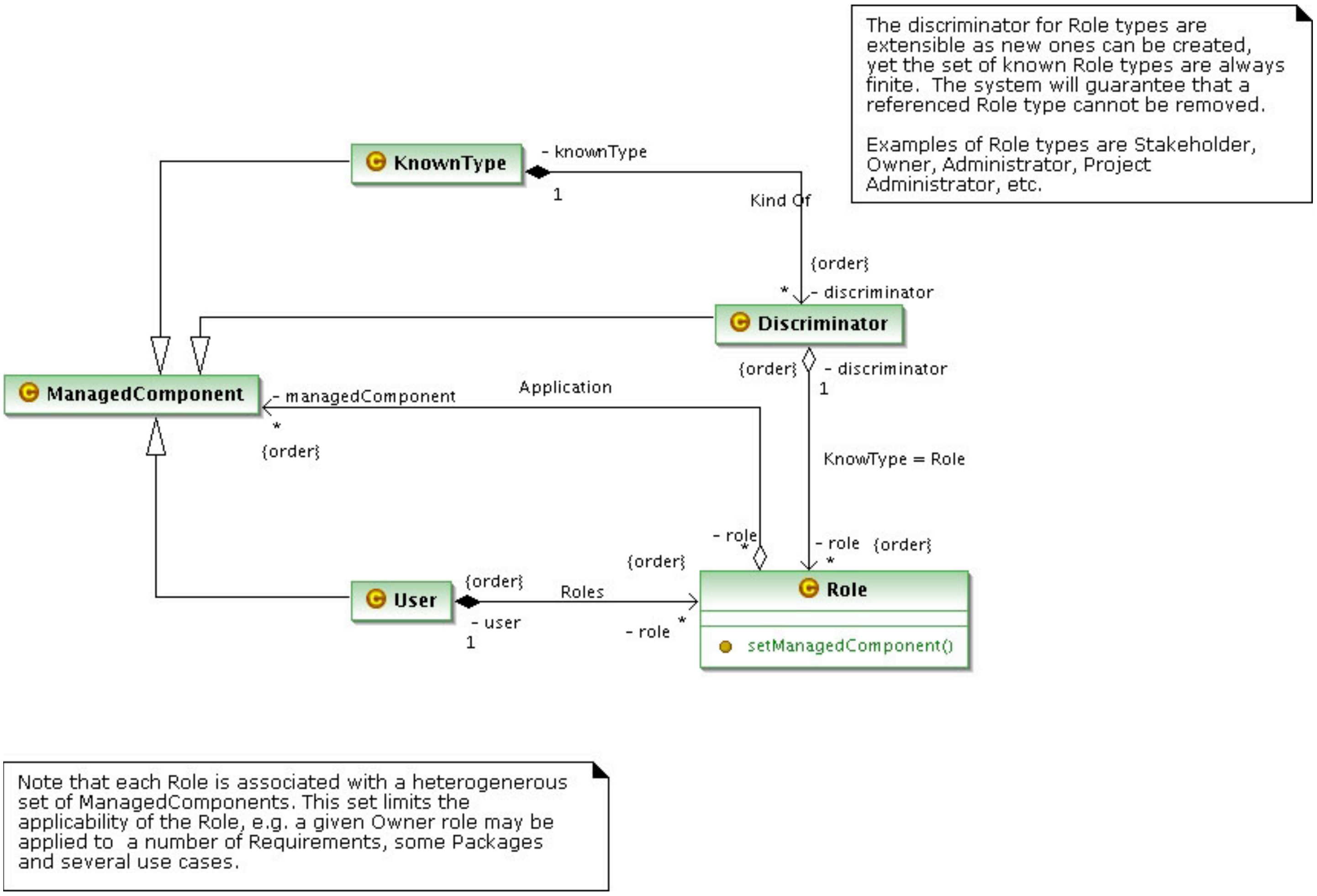
The figure below shows a diagrammatic representation of the concept of Role in the model.

The Role element embodies all of the semantics that is expected to be associated with the role concept.

As the diagram shows, a Role is only valid in the context of the ManagedComponent with which the User is interacting. Consequently, a User may possess many different Role objects for different ManagedComponents.

Each instance of the Role element must be associated with a Discriminator for KnownType = Role that defines the specific role type for that instance of Role.

ORMF comes with a number of pre-defined role types, but the list may be easily extended using the pattern described in section KnownType/Discriminator pattern.



The discriminator for Role types are extensible as new ones can be created, yet the set of known Role types are always finite. The system will guarantee that a referenced Role type cannot be removed.

Examples of Role types are Stakeholder, Owner, Administrator, Project Administrator, etc.

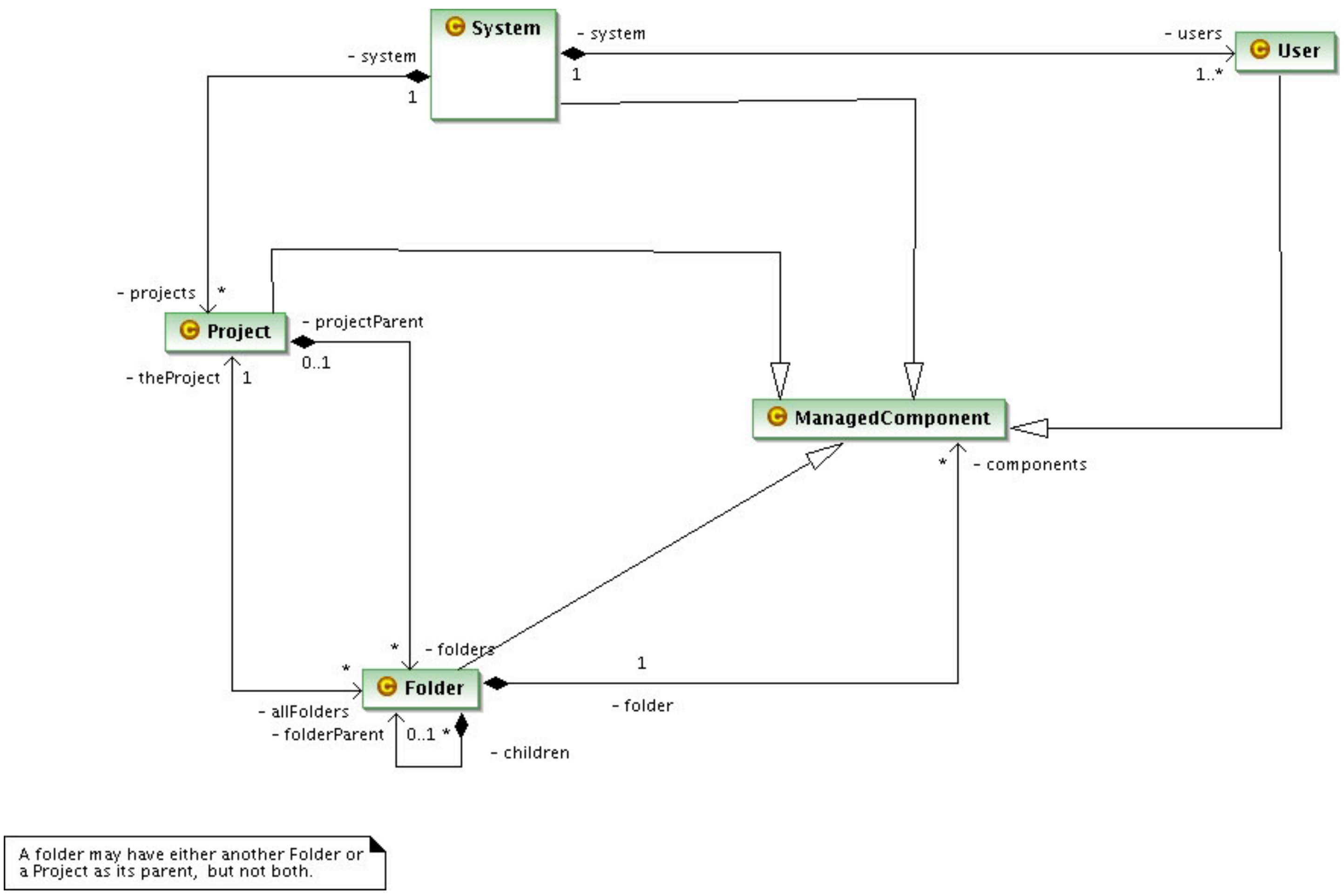
Note that each Role is associated with a heterogenerous set of ManagedComponents. This set limits the applicability of the Role, e.g. a given Owner role may be applied to a number of Requirements, some Packages and several use cases.

System and related elements

The figure below shows the containment relationships between System, User, Project, Folder and ManagedComponent elements.

As already mentioned, the System is at the root of the containment hierarchy. It contains any number of User elements and any number of Project elements. Each Project, in turn, contains a number of Folder elements, each of which may contain either more Folder elements or other ManagedComponent objects.

A Folder always knows about its containing Project, regardless if the Project element is its direct parent or not.



A folder may have either another Folder or a Project as its parent, but not both.

Requirements elements

The figure below focusses on those artefacts that directly represent some form of Requirement.

The basic element embodying a requirement of some type is the RequirementArtifact. This is the most generic requirement related element and it is specialised by two types, namely *Requirement* and *RequirementNarrative*

The *Requirement* element represents such artifacts as a Functional Requirement, a Constraint or a Quality Attribute.

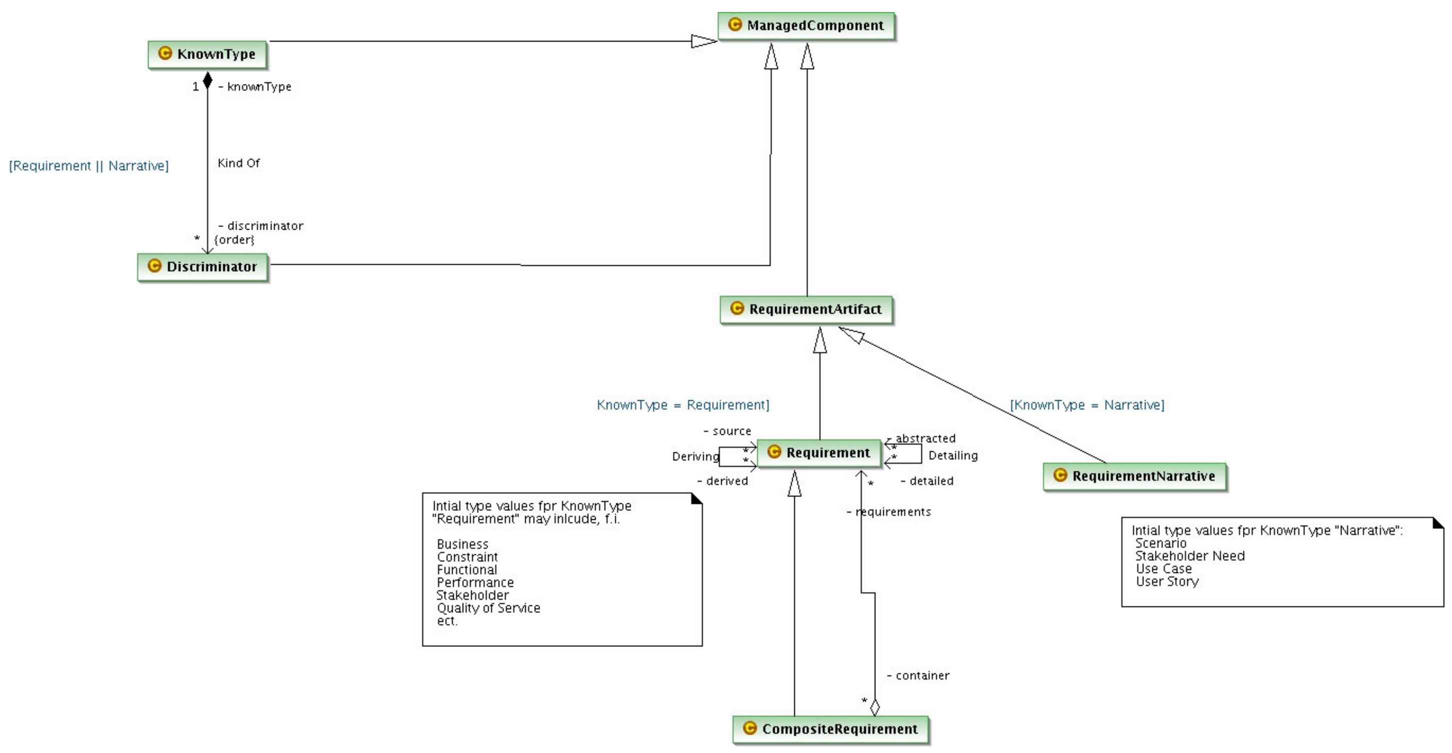
The *RequirementNarrative* elements represents text centric artifacts such as a Use Case, a User Story or a Scenario.

All of the behaviour that is common to requirement artifacts in the categories of Requirement or RequirementNarrative respectively is specified in these two elements.

Requirement and RequirementNarrative take advantage of the KnownType/Discriminator pattern by indicating their specific type via a particular Discriminator associated with KnownType = Requirement or KnownType = Narrative respectively (see section KnownType/Discriminator pattern for details on this pattern).

Discriminators for KnownType = Requirement that are known to ORMF out of the box are inclusive of Business, Constraint, Functional, Performance, Quality of Service and Stakeholder requirement.

Discriminators for KnownType = Narrative that are known to ORMF out of the box are inclusive of Scenario, Stakeholder Need, Use Case and User Story.



Intial type values fpr KnownType "Requirement" may inlcude, f.i.

Business
Constraint
Functional
Performance
Stakeholder
Quality of Service
ect.

Intial type values fpr KnownType "Narrative":
Scenario
Stakeholder Need
Use Case
User Story

Furthermore a Requirement element is characterised by associations with other Requirement elements which can be summarised as defining Requirements that are derived from a source Requirement and/or Requirements that provide greater detail to an abstracted Requirement.

Finally we have the concept of a CompositeRequirement, which is a specialisation of the Requirement element that acts as a container for other Requirement elements.