

Requirements Management Using Enterprise Architect

Enterprise Architect: is a powerful CASE tool for Specifying ,Documenting and Building software project.

Enterprise Architect (EA) is one of the few UML tools that integrate requirements management with other software development disciplines, by creating requirements directly in the model.

Enterprise Architect Features

- ☐ The ability to create and view requirements directly in the model.
- ☐ The ability to detail use cases directly in the model.
- ☐ The ability to enter attributes for each requirement such as difficulty, status, and type; you can also define your own attributes.
- ☐ The ability to trace requirements to business rules, test cases and analysis artifacts.
- ☐ A Relationship Matrix for traceability and viewing the impact of changes to requirements.
- ☐ The ability to create documents.

Requirements Management with UML

Requirements are essentially what the system, application or business process is required to do.

The management of requirements is one of the more difficult and problematic disciplines in the software development industry. The most significant reasons for this are the following:

- ☐ Diverse group input into the requirements
- ☐ Volatility of requirements
- ☐ Imprecision and ambiguities of natural languages

The Enterprise Architect can be used to reduce (and in many circumstances remove) these problems.

Glossary of Terms

The following is a list of terms, and how they relate to requirements management and Enterprise Architect:

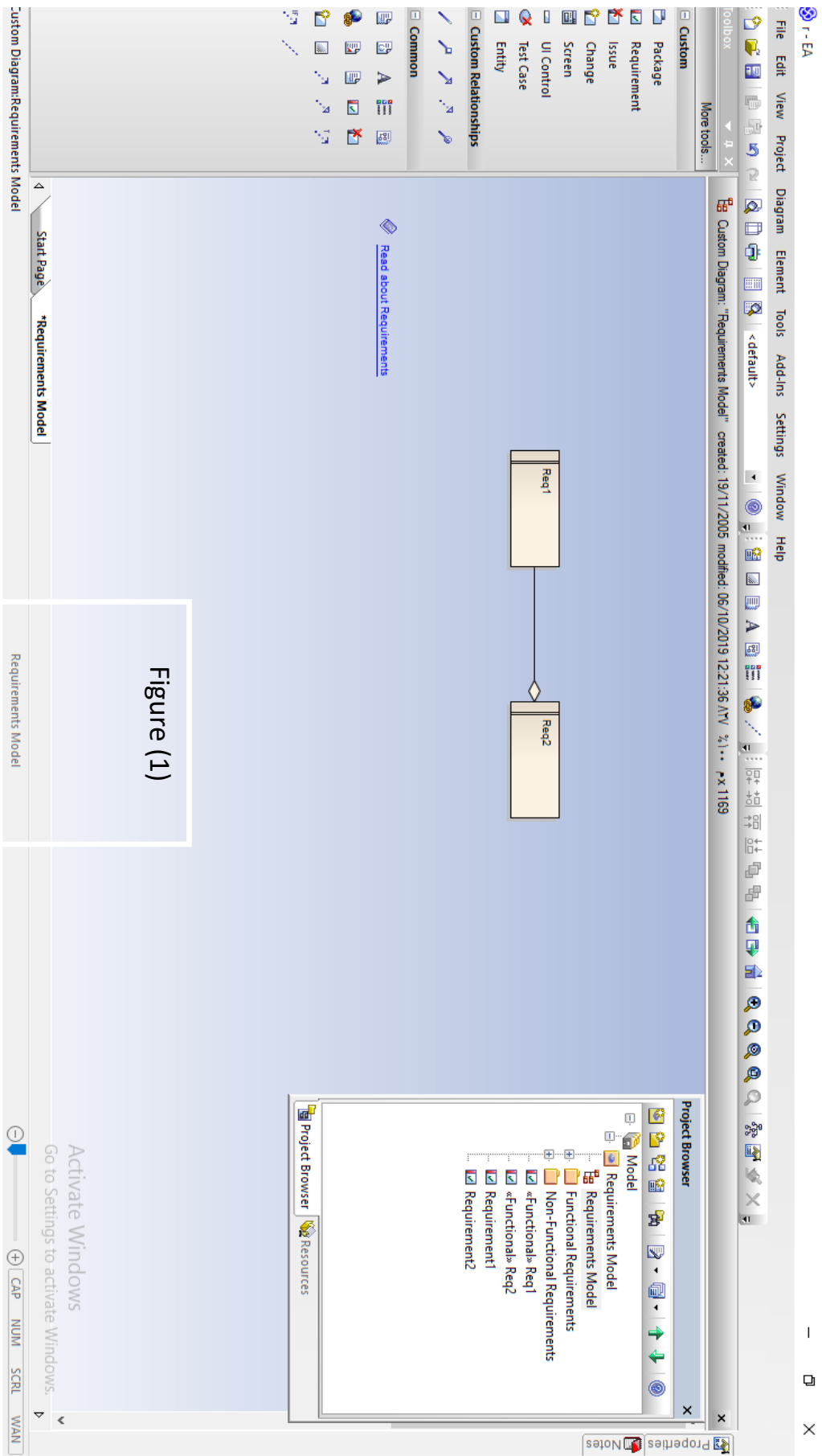
- **Element** – A generic term referring to a singular object in a model. Some of the common elements you will come across include requirements, actors and systems.
- **External requirement** – A requirement that is modeled as an element.
- **Internal requirement** – A requirement that is modeled as the “responsibility” of an existing element.
- **Model** – A representation of a particular system, such as a business process or a database.
- **Diagram** – A common way of representing the way in which models and elements interact.
- **Attributes** – Data fields containing information within requirement elements.

Requirements Modeling

for model requirement, we must use the Requirements palette of the Toolbox that is located at the far left of the Enterprise Architect interface, in the Toolbox as shown in the figure 1.

The contents of the toolbox

Package: is a namespace as well as an element that can be contained in other Package's namespaces. A Package can own or merge with other Packages, and its elements can be imported into a Package's namespace, (Package imports or merges).



Requirement: is a custom element used to capture requirement. A requirement expresses required system behavior. You can connect requirement to other element using realize connector. -٢

Issue: is a structured comment that contains information about defect and issues relating to the system. -٣

Change: is a structured comment that contains information about changes requested to the system. -٤

Screen: is used to prototype User Interface screen flow. you can build up a solid and detailed understanding of user interface behavior without having to use code. It is display a GUI. -٥

UI Control: represents a user interface control element (such as an edit box). It is used for capturing the components of a screen layout and requirements in a Custom or User Interface diagram. -٦

Test Case: Within the Test Case element properties you can define test requirements and constraints, and associate the test with test files. The Test Case element enables you to give greater visibility to tests. -٧

Entity: simple element that represents any general thing. -٨

Relationship Types (Connector Types)

An Association: implies two model elements have a relationship, this connector can include named roles at each end, multiplicity, direction and constraints. **Association:** is the general relationship type between elements. -١

An Aggregation: Requirements linked by aggregation relationships form a composition hierarchy. is a type of association that shows that an element contains or is composed of other elements. It is used to show how more complex elements (aggregates) are built from a collection of simpler elements (component parts; for example, a car from wheels, tires, motor and so on). -٢

A Generalization: is used to indicate inheritance. Drawn from the specific classifier to a general classifier, the generalize implication is that the source inherits the target's characteristics.

A Realizes: a source object implements or Realizes its destination object. Realize connectors are used to express traceability and completeness in the model.

A Nesting: is an alternative graphical notation for expressing containment or nesting of elements within other elements. It is most appropriately used for displaying Package nesting in a Package diagram.

Figure (2) shows the requirements model for Inventory Management.

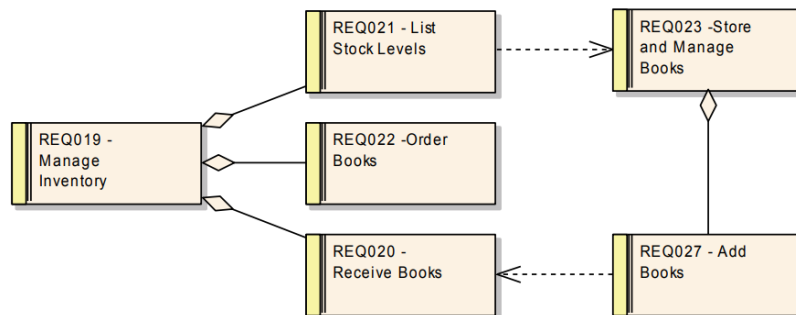


Figure (2)

Figure (3) shows the requirements model for User Account Management

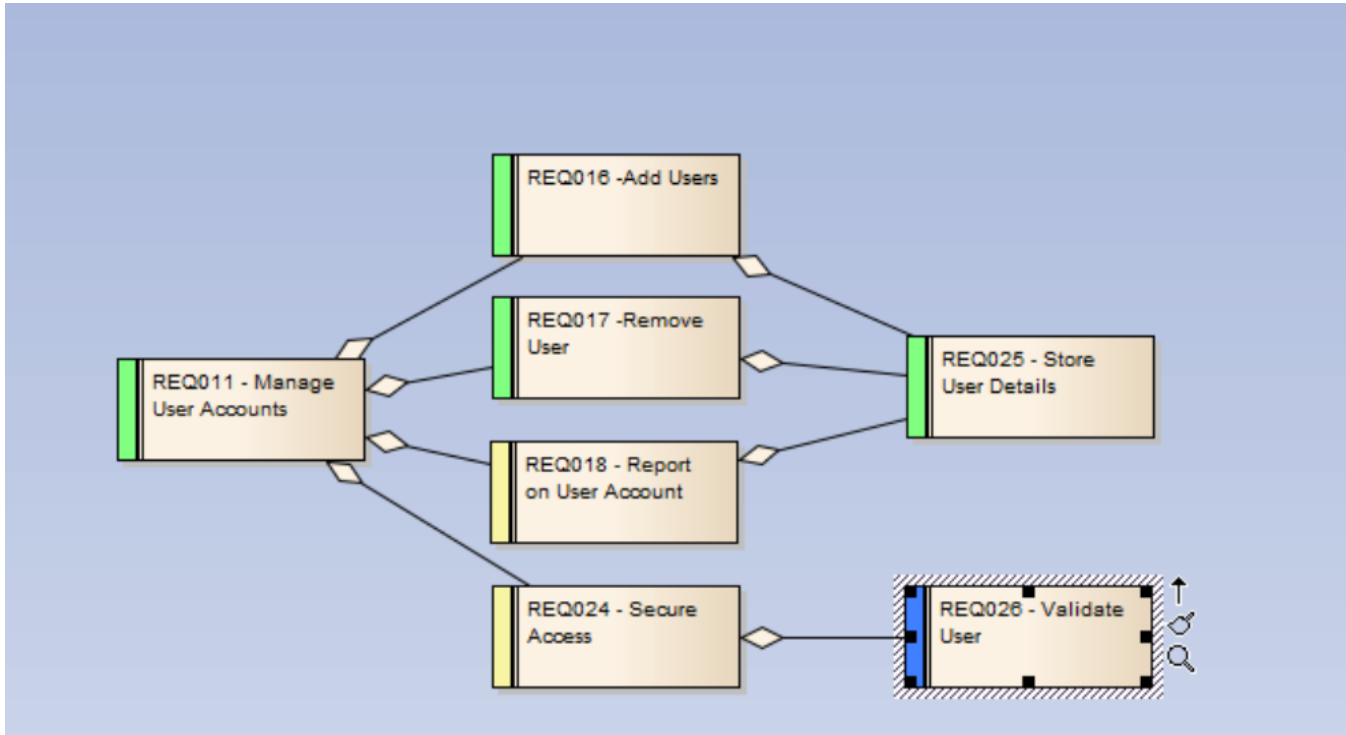


Figure (3)

The two main differences between model and diagram are as follows:

Model can have many different forms, not necessarily graphical. Diagram is - ١
a graphical representation.

Model usually focuses on a broader set of aspects. It is often very complex. - ٢

You look at the system as a whole, so you need to understand all (or most) classes in it, all (or at least or complex) behaviours and so on. Diagrams on the other hand are used specifically to simplify the view. To "slice" it into smaller bits, those are easy to understand.

The mean of source and target of each relationship used in requirements model:





Realize  : -\

 Depend on ----- Needed by


 e.g Computer ----- Power

2- Aggregate  :

 Part of _____ Composed of


Software department  Computer Science College

3- Generalization

 Subtype of _____ Super type of

 Lecturer _____ employee

Nesting  : -ξ

_____  Owned by Owns

—Software Department  Lab 103

Associate  : -o

_____ link from link to