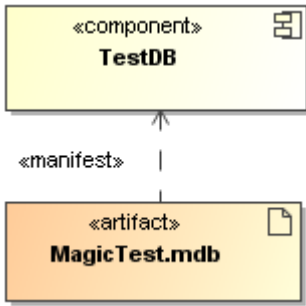
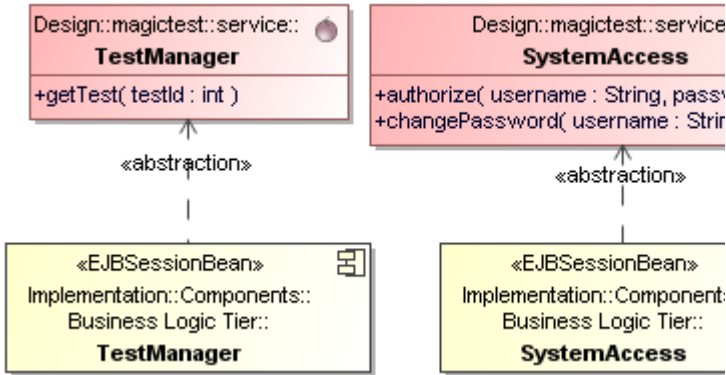
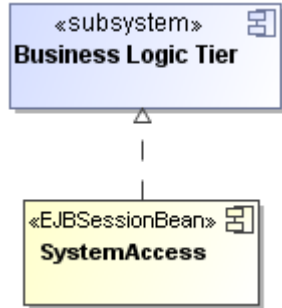
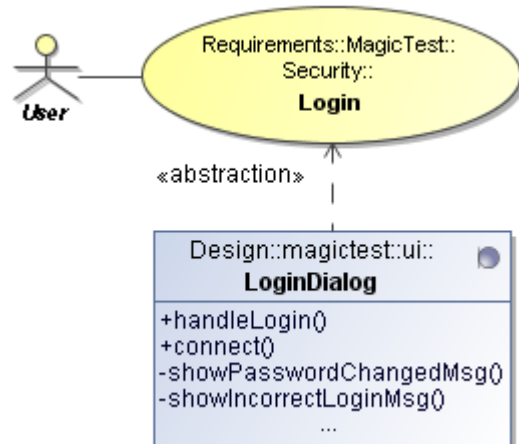


Backward traceability – specification

Property Name	Description	Applied For:	Reference Through:	Value elements type	Example
Manifested in Artifacts	The property shows which components are manifested in an artifact.	Artifact	Relationships: Manifestation	Component	 <pre> graph BT MagicTest.mdb["«artifact» MagicTest.mdb"] -- «manifest» --> TestDB["«component» TestDB"] </pre>
Specifying Class	The property shows the classes representing the component specification from the Design model.	Component	Relationships: Abstraction	Class	 <pre> graph BT subgraph Design TM_Spec["Design::magictest::service:: TestManager +getTest(testid : int)"] SA_Spec["Design::magictest::service:: SystemAccess +authorize(username : String, pass +changePassword(username : Strir"] end subgraph Implementation TM_Impl["«EJBSessionBean» Implementation::Components:: Business Logic Tier:: TestManager"] SA_Impl["«EJBSessionBean» Implementation::Component: Business Logic Tier:: SystemAccess"] end TM_Impl -- «abstraction» --> TM_Spec SA_Impl -- «abstraction» --> SA_Spec </pre>
Specifying Component	The property shows the components that are realized by classifiers through component realization.	Classifier	Relationships: Component Realization, Realization	Component	 <pre> graph BT SystemAccess["«EJBSessionBean» SystemAccess"] -- realization -- > BusinessLogicTier["«subsystem» Business Logic Tier"] </pre>
Specifying Use Case	The property shows the use cases (from the Requirements model) representing the class specification.	Class	Relationships: Abstraction	Use Case	 <pre> graph BT User((User)) --- LoginReq(["Requirements::MagicTest:: Security:: Login"]) LoginDialog["Design::magictest::ui:: LoginDialog +handleLogin() +connect() -showPasswordChangedMsg() -showIncorrectLoginMsg() ..."] -- «abstraction» --> LoginReq </pre>

Specifying Use Case	The property shows the use cases that specify the given use case in the higher level of abstraction. For example, the Business Use Case specifies the Requirements Use Case.	Use Case	Relationships: Abstraction	Use Case	<pre> graph BT Administrate((Administrate)) ModifyUser((Modify User)) RemoveUser((Remove User)) Administrate -.-> «abstraction» ModifyUser Administrate -.-> «abstraction» RemoveUser </pre>
Realized Interface	The property shows the interfaces specifying the contract, in which the related classifier conforms to.	Classifier	Relationships: Interface Realization	Interface	<pre> classDiagram class NotificationService { +send(address : String, message : String) } class NotificationServer { <<component>> } NotificationServer -- > NotificationService </pre>
Specifying Element, All Specifying Elements	<p>The Specifying Element property gathers specifying elements from the upper abstraction level.</p> <p>The All Specifying Elements property transitively gathers specifying elements from all upper abstraction levels.</p> <div style="border: 1px solid orange; padding: 5px; margin-top: 10px;"> <p> Available in Enterprise Architect only.</p> </div>	Element	Relationships: Abstraction, Component Realization, Interface Realization.	Element	<pre> graph TD subgraph Requirements_model [Requirements model] UC[UC] Activity[Activity] Iteration[Iteration] UC --> Activity UC --> Iteration Activity --> Iteration end subgraph Design_model [Design model] Class[Class] end subgraph Implementation_model [Implementation model] Component[Component] Artifact[Artifact] Interface[Interface] Component --> Artifact Component --> Interface Artifact --> Interface end UC -.-> OwnedBehavior Activity UC -.-> OwnedBehavior Iteration Class -.-> Abstraction UC Class -.-> Abstraction Activity Class -.-> Abstraction Iteration Component -.-> Manifestation Class Component -.-> Interface Realization Interface Artifact -.-> Interface Realization Interface </pre>