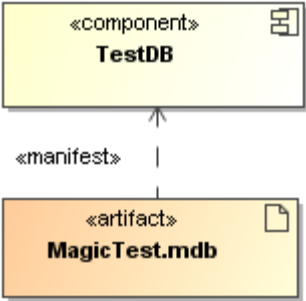
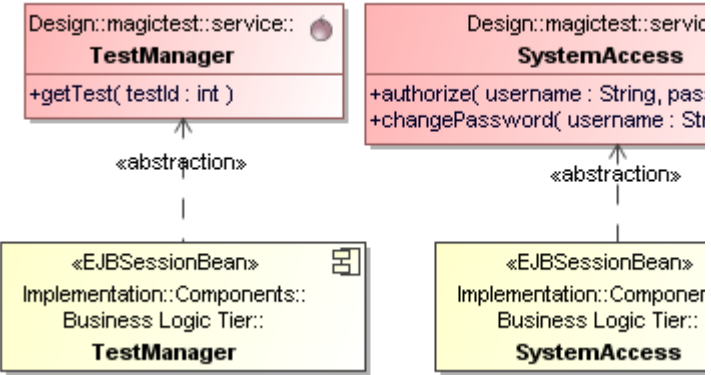
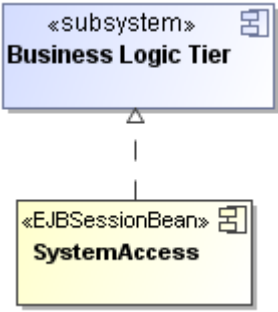
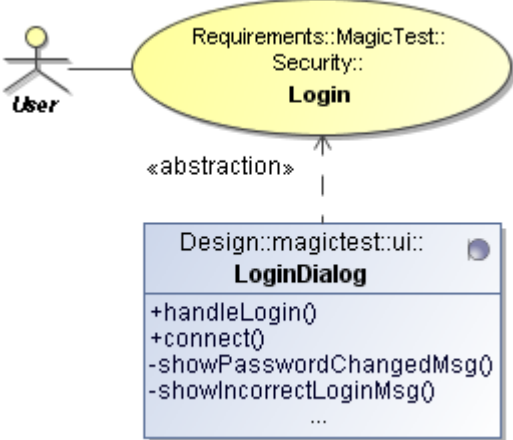


Forward traceability - realization

The forward traceability ensures that all specified artifacts are covered by elements from the lower abstraction level.

Property Name	Description	Applied for	Reference through...	Value elements type	Example
Manifested By Artifacts	The property shows artifacts that physically render the current component.	Component	Relationships: Manifestation	Artifact	 <pre> graph BT TestDB["«component» TestDB"] MagicTest["«artifact» MagicTest.mdb"] TestDB -- «manifest» --> MagicTest </pre>
Realizing Component	The property shows the components representing the class realization in the Implementation model.	Class	Relationships: Abstraction	Component	 <pre> graph BT subgraph Design TM_Svc["Design::magictest::service:: TestManager +getTest(testId : int)"] SA_Svc["Design::magictest::service:: SystemAccess +authorize(username : String, pas +changePassword(username : Str"] end subgraph Implementation TM_EJB["«EJBSessionBean» Implementation::Components:: Business Logic Tier:: TestManager"] SA_EJB["«EJBSessionBean» Implementation::Componer Business Logic Tier:: SystemAccess"] end TM_Svc -- «abstraction» --> TM_EJB SA_Svc -- «abstraction» --> SA_EJB </pre>
Realizing Classifier	The property shows classifiers that realize components through component realization.	Component	Relationships: Component Realization, Realization	Classifier	 <pre> graph BT BLT["«subsystem» Business Logic Tier"] SA_EJB["«EJBSessionBean» SystemAccess"] BLT --> SA_EJB </pre>
Realizing Class	The property shows the classes representing the use case realization in the Design model.	Use Class	Relationships: Abstraction	Class	 <pre> graph BT User((User)) --- LoginReq(["Requirements::MagicTest:: Security:: Login"]) subgraph Design LoginDlg["Design::magictest::ui:: LoginDialog +handleLogin() +connect() -showPasswordChangedMsg() -showIncorrectLoginMsg() ..."] end LoginReq -- «abstraction» --> LoginDlg </pre>

Realizing Use Case	The property shows the realizing use cases of the current use case in the lower level of abstraction thus demonstrating how the use case is implemented. For example, the Requirements Use Case realizes the Business 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>
Realizing Classifier	The property shows the classifiers conforming to the contract specified by the interface and related with this interface through the interface realization relationship.	Interface	Relationships: Interface Realization	Classifier	<pre> classDiagram class NotificationService { +send(address : String, message : String) } class NotificationServer { <<component>> } NotificationServer -- > NotificationService </pre>
Realizing Element, All Realizing Elements.	<p>The Realizing Element property gathers realizing elements from the lower abstraction level.</p> <p>The All Realizing Elements property transitively gathers realizing elements from all lower abstraction levels.</p>	Element	Relationships: Abstraction, Component Realization, Interface Realization.	Element	<pre> graph TD subgraph Requirements_model [Requirements model] UC[UC] Activity[Activity] Iteration[Iteration] end subgraph Design_model [Design model] Class[Class] end subgraph Implementation_model [Implementation model] Component[Component] Artifact[Artifact] Interface[Interface] end UC --> Activity UC --> Iteration Activity --> Iteration Class --> Component Component --> Artifact Component --> Interface Artifact --> Interface UC -.-> OwnedBehavior Activity UC -.-> OwnedBehavior Iteration Activity -.-> OwnedBehavior Iteration Class -.-> Abstraction Component Class -.-> Abstraction Artifact Class -.-> Abstraction Interface Component -.-> Manifestation Artifact Component -.-> Manifestation Interface Artifact -.-> Interface Realization Interface </pre>