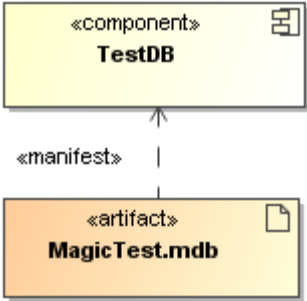
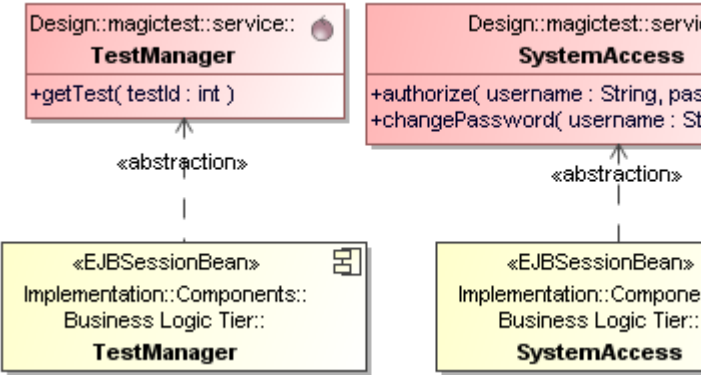
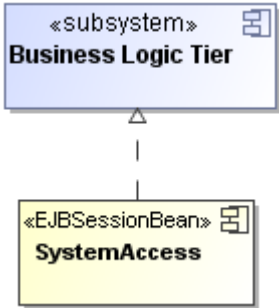
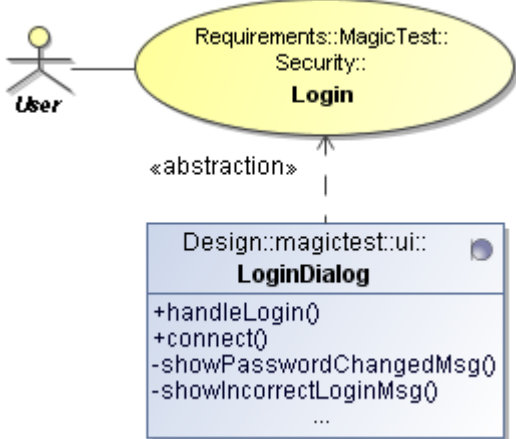


Forward traceability - realization

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 MagicTest.mdb["«artifact» MagicTest.mdb"] -- «manifest» --> TestDB["«component» TestDB"] </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 : St"] end subgraph Implementation TM_EJB["«EJBSessionBean» Implementation::Components:: Business Logic Tier:: TestManager"] SA_EJB["«EJBSessionBean» Implementation::Compone Business Logic Tier:: SystemAccess"] end TM_EJB -- «abstraction» --> TM_Svc SA_EJB -- «abstraction» --> SA_Svc </pre>
Realizing Classifier	The property shows classifiers that realize components through component realization.	Component	Relationships: Component Realization, Realization	Classifier	 <pre> graph BT SA_EJB["«EJBSessionBean» SystemAccess"] -- realization --> BLT["«subsystem» Business Logic Tier"] </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)) --- Login_UseCase(["Requirements::MagicTest:: Security:: Login"]) LoginDialog["Design::magictest::ui:: LoginDialog +handleLogin() +connect() -showPasswordChangedMsg() -showIncorrectLoginMsg() ..."] -- «abstraction» --> Login_UseCase </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.	The Realizing Element property gathers the source elements of the abstraction relationship. The All Realizing Elements property gathers recursively (performs a realizing element search for the result) source elements of the abstraction relationship. Available in Enterprise Architect only.	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 end subgraph Design_model [Design model] Class[Class] end subgraph Implementation_model [Implementation model] Component[Component] Artifact[Artifact] Interface[Interface] Component --> Artifact Component --> Interface end UC -.-> OwnedBehavior Activity UC -.-> OwnedBehavior Iteration Class -.-> Abstraction UC Component -.-> Manifestation Class Component -.-> Interface Realization Interface </pre>