Source π-UseCase meta-model	Mapping Rules	Target π-ServiceProcess meta-model
Constraint	- All **Constraints** in the source model will be transformed into a **Assertion** in the target model. All **Use Cases** are associated to a **Non-Functional Requirement** and all **Constraints** are associated with a **Non-Functional Attribute**. Thus, **Constraint** representing the same Use Case, is transformed in a **Contract**. - A **Contract** can also have a exceptional cases. The negation of the **Constraint** predicate are treated as the type *EXCEPTIONALBEHAVIOUR*. Thus, all the **Assertions** related to this **Constraint**, in this particular case, will be transformed in a **Exceptional Behaviour**. - The set of ASSERTIONS of a USE CASE are associated with a CONTRACT - The CONTRACT name is formed by "<useCaseName>" + "Contract" token.	Assertion, Contract, Exceptional Behaviour
Constraint Type	- The **Constraint Type** in source model can have the following transformation: • **BUSINESS** type is transformed into none **Assertion**; • **VALUE** with the **Assertion** *isExceptionalBehaviour* attribute setted to false is transformed into a **Assertion** in the target model; • **EXCEPTIONALBERAVIOR** with the **Assertion** *isExceptionalBehaviour* attribute setted to true is transformed into a **Exceptional Behaviour** in the target model. - Both, the value and exceptional restriction are based on in accordance with the values of **Assertions** - **Constraint Type** defines the type of restriction on a **Use Case**.	Assertion Exceptional Behaviour
Non-Functional Attribute	Every **Non-Functional Attribute** associated to an element (**Constraint** and **Non-Functional Requirements**) in the source model becomes a **Non-Functional Attribute** associated to the corresponding element (**Contract**) in the target model.	Non-Functional Attribute
Use Case	For every **Use Case** there will be a **Action** in the service delivery process model describing the Business Service	Action
Extend (Use Case)	- Every **Extend** association identified in the *π-UseCase model* will be represented in the target model by a **Fork Node**. - The **Service Activity** corresponding to the source **Use Case** of the extend association must be previous to the **Service Activity** corresponding to the target **Use Case** of the extend association. - If the extend association has only one source **Use Case**, the fork will present the **Service Activity** as an alternative to another flow with no **Service Activity**. Later, both flows will meet; - If the extend association has several source **Use Case**, the fork will present the different **Service Activities** as mutual alternatives to another flow with no **Service Activity**. Later, all these flows will meet;	Control Node, Fork Node, Service Activity
Include (Use Case)	- An **Include** association is found in the *π-UseCase model*, the **Service Activity** corresponding to the source must be subsequent to the **Service Activity** corresponding to the target **Use Case** of the include association; - If the include association has several targets, the designer must decide the appropriate sequence for the different **Service Activities** corresponding to the target **Use Case** (which will obviously be previous to the **Service Activity** corresponding to the source **Use Case**).	Service Activity
Requirement	- A **Requirement** in the source model is transformed into **Service Activity** of the target model. - The rules for a **Requirement** to be transformed into a service activity is analyzed after the transformations of included and extended use cases. - A **Requirement** associated with only one **Use Case** in the source model is transformed in a. **Service Activity** and an **Action**, respectively. - All **Use Cases** must be associated with a **Requirement**.	Service Activity