Transformation AML-Model zu Code

Umsetzung für Besprechung am 09.08.2017.

Für AMLmodel\_v3.

Inhalt

[ArchitectureRoleClassLib/Service 2](#_Toc489720218)

[ArchitectureRoleClassLib/Component 3](#_Toc489720219)

[CommunicationRoleClassLib/Ports 4](#_Toc489720220)

[TODO: CommunicationRoleClassLib/Port 5](#_Toc489720221)

[TODO: DomainModelRoleClassLib/MessageModel 6](#_Toc489720222)

[AML-Elemente ohne Auswirkung auf den generierten Code 7](#_Toc489720223)

[TODO: Festlegungen und bekannte Einschränkungen 8](#_Toc489720224)

[betreffend das AML-Model 8](#_Toc489720225)

[OLD: Festlegungen 8](#_Toc489720226)

[OLD: Einschränkungen 8](#_Toc489720227)

[mögliche Verbesserungen 8](#_Toc489720228)

# ArchitectureRoleClassLib/Service

|  |  |
| --- | --- |
| * Für jedes direkte IE mit Role ArchitectureRoleClassLib/Service in der Instance-Hierarchy:   + neues Maven-Projekt im Codegenerator-Output-Directory in Subfolder <IE-Name> anlegen:     - LICENCE.txt anlegen 🡪 leeres TXT     - NOTICE.txt anlegen 🡪 leeres TXT     - README.txt anlegen 🡪 leeres TXT     - pom.xml anlegen 🡪 ‚groupId’ und ‚artifactId’ wie in den Attributen des IE spezifiziert setzen, ‚name’ = <IE-Name>, initial keine ‚modules’ | |
|  |  |

# ArchitectureRoleClassLib/Component

|  |  |
| --- | --- |
| * Für jedes direkte IE mit Role ArchitectureRoleClassLib/Component unterhalb eines IE mit Role ArchitectureRoleClassLib/Service:   + neues Maven-Module mit Namen <Component-artifactId> anlegen im Maven-Projekt des Services:     - Main.java anlegen 🡪 Klasse mit Methode ‚main’, in der ‚// TODO: implement me’ steht     - log4j2.xml anlegen 🡪 Standard-Logging-Konfiguration     - pom.xml anlegen 🡪 ‚parent’: den Service setzen, ‚groupId’ und ‚artifactId’ setzen, initial keine ‚dependencies’, ‚profile’ anlegen, welches die Main.java ausführt   + pom.xml von Service anpassen 🡪 module <Component-artifactId> hinzufügen | |
|  |  |

# CommunicationRoleClassLib/Ports

|  |  |
| --- | --- |
| * Für jedes direkte IE mit Role CommunicationRoleClassLib/Ports unterhalb eines IE mit Role ArchitectureRoleClassLib/Component:   + wenn Maven-Module ‚communication’ noch nicht angelegt ist:     - neues Maven-Module mit Namen ‚communication’ anlegen und die fixfertigen Komponenten darin hinzufügen     - pom.xml von Service anpassen 🡪 module ‚communication’ hinzufügen   + Maven-Module der Component anpassen:     - CommunicationService.java anlegen 🡪 hat initial noch keine Ports zugewiesen     - Main.java anpassen 🡪 Example-Usage des CommunicationService der Methode ‚main’ hinzufügen     - pom.xml anpassen 🡪 module ‚communication’ als ‚dependency’ hinzufügen | |
|  |  |

# TODO: CommunicationRoleClassLib/Port

|  |  |
| --- | --- |
| * Für jedes direkte IE mit Role CommunicationRoleClassLib/Port unterhalb eines IE mit Role CommunicationRoleClassLib /Ports:   + Maven-Module der Component anpassen:     - CommunicationService.java anpassen 🡪 Port zu enum(s) hinzufügen und Port in Methode ‚init’ einrichten | |
|  |  |

# TODO: DomainModelRoleClassLib/MessageModel

|  |  |
| --- | --- |
| * Wenn das IE mit Role CommunicationRoleClassLib/Port ein zugewiesenes IE mit Role CommunicationRoleClassLib /MessageModel hat:   + wenn Maven-Modul ‚messagemodel’ noch nicht angelegt ist:     - neues Maven-Module mit Namen ‚messagemodel’ anlegen und fixfertige MessageModel.java darin hinzufügen     - pom.xml von Service anpassen 🡪 ‚module’ ‚messagemodel’ hinzufügen   + wenn pom.xml von Component noch keine ‚dependency’ namens ‚messagemodel’ hat:     - pom.xml von Component anpassen 🡪 ‚dependency’ ‚messagemodel’ hinzufügen   + wenn spezifiziertes Message-Model in Maven-Module ‚messagemodel’ noch nicht angelegt ist:     - spezifiziertes Message-Model in Maven-Module ‚messagemodel’ anlegen | |
|  |  |

# AML-Elemente ohne Auswirkung auf den generierten Code

Folgende RoleClassLibs haben derzeit keine Auswirkung auf den generierten Code:

* EnvironmentRoleClassLib
* ProgrammingLanguageRoleClassLib
* SourcecodeRepositoryRoleClassLib

# TODO: Festlegungen und bekannte Einschränkungen

## betreffend das AML-Model

### OLD: Festlegungen

* alle IEs mit Role ArchitectureRoleClassLib/Component
  + müssen unique Names haben (da diese im Code verwendet werden)
  + dürfen nur ein Kind-IE mit Role CommunicationRoleClassLib/Ports haben
    - dessen Kind-IEs mit Role CommunicationRoleClassLib/Port
      * einen unique Name besitzen (nicht zwingend nötig, macht aber Sinn, um Verwirrungen vorzubeugen)
      * Message-Models sind optional
      * nur ein EI mit Namen ConnectionPoint haben dürfen
        + muss IL zu einem IE mit Role ArchitectureRoleClassLib/Component haben und nicht zu sich selbst
        + der RefBaseClassPath auf CommunicationInterfaceClassLib/Port/Sender oder CommunicationInterfaceClassLib/Port/Receiver gesetzt hat

Gegenstelle muss immer entsprechender Gegenpart sein (Sender – Receiver oder Receiver – Sender aber nicht Sender – Sender oder Receiver – Receiver)

### OLD: Einschränkungen

* Code kann derzeit nur generiert werden, wenn alle IEs mit Role ArchitectureRoleClassLib/Component eine Role ProgrammingLanguageRoleClassLib/Java zugewiesen haben
* Kommunikationsart (RMI etc.) wird derzeit für alle Ports festgelegt, individuell pro Port ist nicht möglich

### mögliche Verbesserungen

* derzeit werden alle Port-Factory- und Port-Implementierungen in das Maven-Modul ‚communication’ kopiert. Hier würde es ausreichen, nur die Implementierungen zu kopieren, die auch tatsächlich im AML-Modell referenziert werden.