Auswirkungen des App Store Templates auf D°

- Template-Applikation
  - Application Logic
  - Infomodel
  - Swagger UI
  - Status
  - Config
  - Policies
- D°-Applikation
  - Application Logic
  - Config
  - Policies
- Unterschiede
- Analyse
  - Application Logic
  - Infomodell
  - Swagger UI
  - Status
  - Config
  - Policies

Versionen

Datum	Notiz	Version
31.03.2021	Erste Veröffentlichung	1.0

Autoren

Name	Institution	Email
Fabian Bruckner	Fraunhofer ISST	fabian.bruckner@isst.fraunhofer.de

Im Rahmen der Entwicklung des App Stores wurde ein Template für Applikationen veröffentlicht, welches eine Möglichkeit beschreibt, wie Applikationen gestaltet werden können, um anschließend im App Store verwendet zu werden. Dabei ist die Umsetzung des Templates nicht verpflichtend, um Applikationen über den App Store zu verbreiten, sondern stellt nur eine Möglichkeit/Anregung zur Verfügung.

Das besagte App-Template wird im vorliegenden Dokument untersucht, um zum einen Unterschiede zu Applikationen, welche mit dem aktuellen Entwicklungsstand von D° erzeugt werden, zu identifizieren und zum anderen sinnvolle Anpassungen & Erweiterungen für D° aufzuzeigen. Dabei geschieht die Analyse auf Basis von verschiedenen Ressourcen.

- App Store Jive Dokuments (https://industrialdataspace.jiveon.com/docs/DOC-2604)
- App Template Repository (https://gitlab.cc-asp.fraunhofer.de/fhg-fit-ids/ids-app-template)

Dabei werden zunächst das Template und D° Applikationen näher betrachtet und anschließend Unterschiede aufgezeigt und gegebenenfalls Änderungen an D° eingeführt.

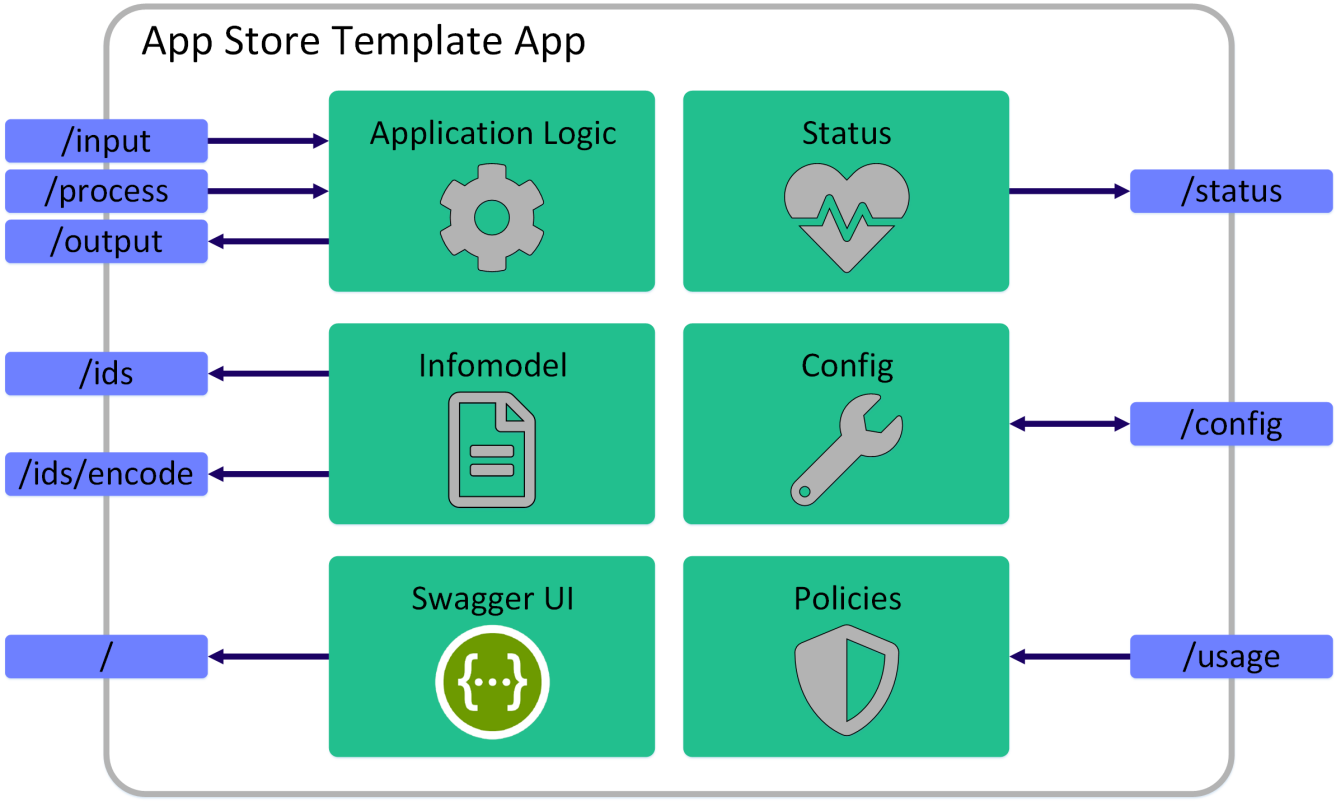
Template-Applikation

Das Template beschreibt Applikationen anhand ihrer Schnittstelle. Dabei stellen Applikationen eine HTTP-Schnittstelle bereit. Die nachfolgende Tabelle kombiniert die verschiedenen verfügbaren Ressourcen, um eine umfängliche Übersicht über die vom Template beschriebene Schnittstelle zu erhalten.

Endpoint	HTTP-Method	Description
App Endpoints		
http://localhost:8080/status	GET	App endpoint for status information (e.g. lifecycle, heartbeat)
http://localhost:8080/input	POST	App endpoint for consuming data to process
http://localhost:8080/output	GET	App endpoint for providing processed data
http://localhost:8080/config	GET/POST	App endpoint for configuration / parameterization
http://localhost:8080/process	POST	App endpoint to start the processing
http://localhost:8080/usage	POST	App endpoint for Usage Control limitations
IDS Endpoints		

http://localhost:8080/ids	GET	Endpoint that provides the app's ids information model representation
http://localhost:8080/ids/encode	GET	Endpoint that provides the app's ids information model representation encoded as base64
Documentation Endpoint		
http://localhost:8080/	GET	Swagger UI documentation for the example app

Für eine einfachere Verständlichkeit wurden die Informationen aus der Tabelle in der nachfolgenden Abbildung zusammengeführt.



Nachfolgend werden die einzelnen Komponenten etwas ausführlicher betrachtet.

Application Logic

Zur Interaktion mit der eigentlichen Applikationslogik stehen drei verschiedene Endpunkte zur Verfügung. Über den `/input`-Endpunkt werden die Eingabeparameter für die Ausführung der Applikationslogik übergeben. Der `/process`-Endpunkt wird verwendet, um die eigentliche Ausführung der Applikationslogik zu starten und zu stoppen. Dabei wird ein boolescher Wert an den Endpunkt übergeben, welcher entscheidet, ob die Ausführung gestartet (`true`) oder gestoppt (`false`) werden soll. Über den `/output`-Endpunkt werden abschließend die Ergebnisse der Applikationslogik abgefragt.

Infomodel

Zwei Endpunkte erlauben es die Metadaten, welche die Applikation beschreiben und im IDS Infomodel vorliegen, abzufragen. Dabei können die Metadaten sowohl im `JSON`-Format (`/ids`), als auch `Base64`-kodiert (`/ids/encode`) abgefragt werden.

Swagger UI

Über die Wurzel der App (`/`) kann eine Swagger Oberfläche erreicht werden. Diese wird (zumindest in dem vorliegenden Template) automatisch aus Annotationen, welche an den einzelnen Parametern und Endpunkt-Methoden angebracht sind, erzeugt.

Status

Über den Endpunkt `/status` können nicht näher benannte Statusinformationen über die App abgefragt werden.

Config

Falls die App Möglichkeiten zur Konfiguration/Parametrisierung bietet, werden die entsprechenden Daten an den `/config`-Endpoint gesendet. Darüber hinaus erlaubt es dieser Endpunkt die aktuelle Konfiguration der App abzufragen.

Policies

Der Endpunkt `/usage` erlaubt es die Policies einer Applikation zu setzen. Diese werden (im Template) als String übergeben und gespeichert und entsprechen nicht der Infomodel Repräsentation von Policies.

An dieser Stelle sei besonders hervorgehoben, dass das Template keine Möglichkeit vorsieht die Policies einer App abzufragen. Es ist nur möglich diese zu setzen.

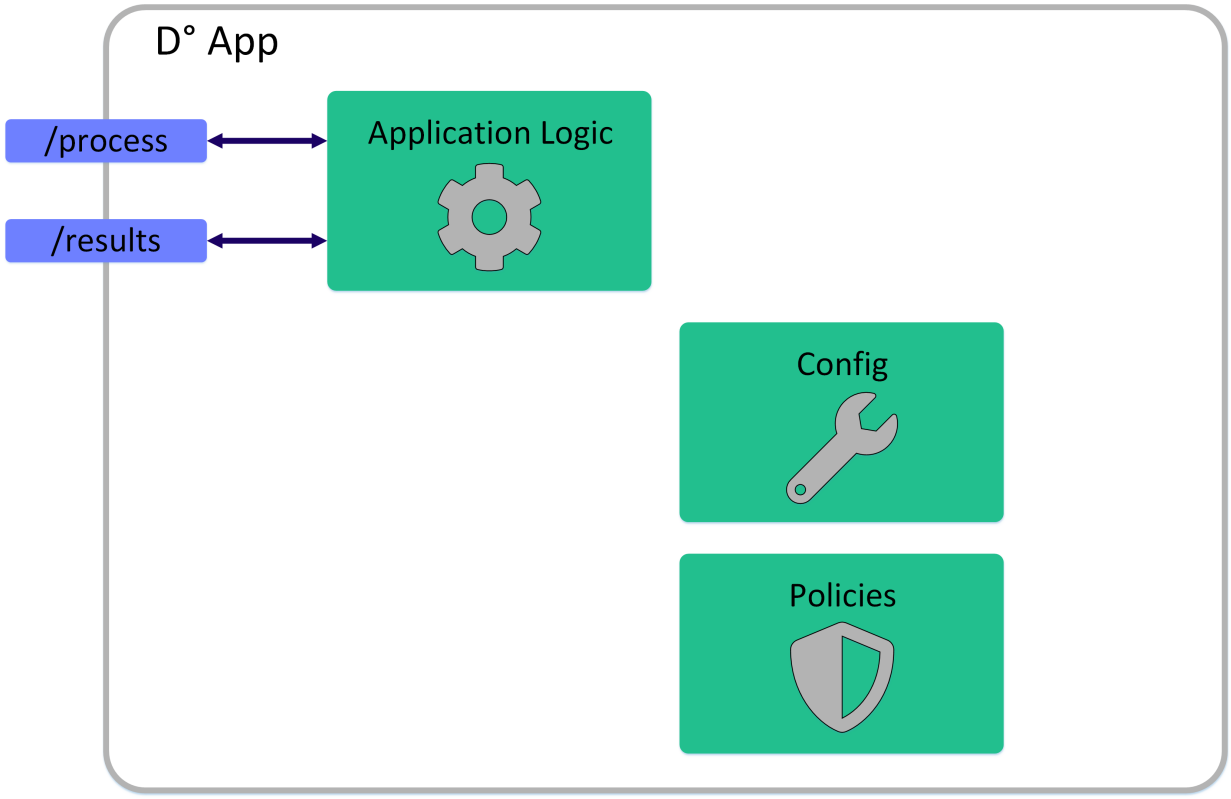
D°-Applikation

Demgegenüber verfügen Applikationen, die mit D° erzeugt werden über eine kompaktere und auch funktional anders aufgebaute Schnittstelle. Auch wenn D° es erlaubt verschiedene Arten von Applikationen zu erzeugen, beschränkt sich dieses Dokument auf solche, die eine HTTP-Schnittstelle bereitstellen, da ansonsten die Vergleichbarkeit nicht gegeben ist.

Die nachfolgende Tabelle listet die Schnittstelle einer D°-Applikation, welche mit dem aktuellen Entwicklungsstand gebaut wurde.

Endpoint	HTTP-Method	Description
http://localhost:8080/process	POST	Starte die Ausführung der Applikationslogik mit gegebenen Eingabeparametern. Gibt eine UUID zurück, welche es ermöglicht diese spezielle Ausführung zu identifizieren.
http://localhost:8080/results	POST	Erlaubt es die Berechnungsergebnisse der Ausführung abzufragen, welche mit einer übergebenen UUID identifiziert wird.

Zum einfacheren Vergleich wurde auch für eine solche D°-Applikation eine Abbildung erzeugt, welche nachfolgend zu sehen ist. Um einen einfachen und schnellen Vergleich der beiden Applikationen zu ermöglichen, wurde darauf verzichtet die Abbildung in Form und Größe zu verändern.



Auch für die D°-Applikation werden die einzelnen Komponenten kurz beschrieben.

Application Logic

Über zwei Endpunkte wird die Applikationslogik gesteuert. Der Endpunkt `/process` erlaubt es die Ausführung der Applikationslogik mit übergebenen Eingabeparametern zu starten. Der Aufrufer erhält dabei als Antwort eine UUID zurück, welche die ausgelöste Ausführung identifiziert. Diese UUID muss an den `/results`-Endpunkt übergeben werden, um die erzeugten Ergebnisse der Ausführung abzufragen.

Config

Die Konfiguration einer D°-Applikation ist ein fester Bestandteil der Applikation, welcher zur Übersetzungszeit in die ausführbare Applikation integriert wird. Dabei ist die Konfiguration weder lesend noch schreibend exponiert.

Policies

Abweichend von der Abbildung sind die Policies einer D°-Applikation keine losgelöste Komponente, welche in der Applikation liegen. Stattdessen sind die einzelnen Policies jeweils an den relevanten Sprachelementen, welche in der Applikation verwendet werden angebracht und untrennbar mit diesen verbunden. Die Vereinigung von Sprachelementen und verwendeten Policies wird während der Übersetzungszeit durchgeführt und kann zur Laufzeit der Applikation nicht mehr geändert werden. Hieraus ergibt sich direkt, dass eine Schreibende Schnittstelle für die Policies in einer D°-Applikation nicht möglich ist. Es gibt aber auch keine Möglichkeiten die Policies abzufragen.

Unterschiede

Es fällt direkt auf, dass die App Store Template App über mehr Komponenten und eine umfangreichere API verfügt. Die D°-Applikation verfügt über keine Swagger-, Infomodel- und Status-Komponente. Dementsprechend entfallen auch die entsprechenden API-Endpunkte für diese Komponenten. Aber auch bei den gemeinsamen Komponenten gibt es Unterschiede in der Schnittstelle. D°-Applikationen verfügen über keine Schnittstellen für die Policies- und Config-Komponente. Dies liegt zu Teilen an dem internen Aufbau von D°-Applikationen begründet.

Aber auch die Interaktion mit der Applikationslogik gestaltet sich bei den beiden Applikationstypen unterschiedlich. Die App Store Template App verwendet unterschiedliche Endpunkte zum setzten der Eingaben, abfragen der Ergebnisse und zum starten/stoppen der Applikation. Hieraus ergibt sich, dass keine mehrfach parallele Ausführung der Applikationslogik möglich ist und das die Berechnungsergebnisse nach jeder Ausführung abgeholt werden müssen, da ansonsten an Verlust droht.

D°-Apps arbeiten mit UUIDs, um individuelle Ausführungen zu identifizieren und erlauben, bei entsprechender Implementation der verwendeten Policies und Aktivitäten, mehrere gleichzeitige Ausführungen der Applikationslogik. Ebenso können die Berechnungsergebnisse einer Ausführung zu einem beliebigen Zeitpunkt abgefragt werden, da hier kein Überschreiben durch andere Ausführungen droht. Dafür bietet D° keine Möglichkeit eine laufende Ausführung der Applikationslogik abzubrechen.

Analyse

In diesem Abschnitt des Dokuments werden die einzelnen Unterschiede zwischen den beiden Applikationstypen genauer untersucht und dabei Mögliche Anpassungen für D° diskutiert. Dabei wird sich strukturell an den Komponenten der App Store Template App orientiert, da die dort verfügbaren Komponenten eine Obermenge der Komponenten der D° Applikation bilden.

Application Logic

Vergleicht man die Schnittstellen für die Applikationslogik der beiden Applikationsarten, ist die D° Variante ein Stück weit flexibler im Hinblick auf gleichzeitig parallele Ausführungen und dem Abrufen von Berechnungsergebnissen. Demgegenüber erlaubt es die App Store Template App eine laufende Ausführung abzubrechen, was in D° so nicht vorgesehen bzw. möglich ist.

Die Möglichkeit eine laufende Ausführung der Applikationslogik könnte einen Mehrwert für D°-Applikationen darstellen, würde aber einen größeren Umbau nötig machen. Zum einen müsste die Codegenerierung dahingehend umgestellt werden, dass nach der Ausführung jeder Aktivität überprüft wird, ob ein Abbruch der Ausführung gefordert ist. Da Aktivitäten aus der Sicht von D° die atomaren Bausteine sind, sind die einzigen validen Abbruchpunkte vor/nach der Ausführung einer Aktivität, niemals während der Ausführung.

Darüber hinaus müsste eine Möglichkeit zum Rückrollen der Teilweise ausgeführten Applikationslogik geschaffen werden. Eine Möglichkeit einzelne Aktivitäten nach ihrer Ausführung zurückzurollen ist in Planung, aber noch nicht umgesetzt. Dies würde es erfordern die Schnittstelle von Aktivitäten dahingehend zu modifizieren, dass bei der Ausführung der Aktivität zunächst nur auf Datenkopien gearbeitet wird und nach erfolgreicher Ausführung der gesamten Applikation alle durchgeführten Änderungen bestätigt und damit dauerhaft gültig werden.

Dabei können sich komplizierte Sonderfälle ergeben, bspw. bei der Kommunikation zwischen Aktivitäten und externen Systemen. Eine andere Möglichkeit wäre es eine laufende Ausführung ohne ein Zurückrollen abzubrechen, was aber zu undefinierten Zuständen und Verhalten führen kann, weswegen hiervon abzusehen ist.

Aus diesen Gründen wird zeitnah keine Möglichkeit für D°-Applikationen geschaffen, welche es erlaubt eine laufende Ausführung der Applikationslogik abzubrechen. Daraus ergibt sich, dass für die Komponente der Applikationslogik keine Anpassungen an D° vorgenommen wird.

Infomodell

Eine Metadatenbeschreibung im Format des IDS Infomodels von einer laufenden Applikation abfragen zu können ist eine nützliche Funktionalität, welche von D°-Applikationen aktuell nicht geboten wird. Die App Store Template App erlaubt es dagegen die Metadaten in verschiedenen Kodierungen abzufragen. Eine Funktionalität, welche den Abruf dieser Daten für D°-Applikationen erlaubt wird zu D° hinzugefügt. Dabei wird aber auf die Auswahl verschiedener Formate verzichtet, weil kein entsprechender Bedarf bekannt ist. Es wird ermöglicht die IDS Metadaten einer Applikation im JSON-Format abzufragen.

Dabei gibt es verschiedene Möglichkeiten zur Umsetzung dieser Funktionalität.

1. Es wird eine statische Datei in die Applikation integriert, welche bei Bedarf ausgeliefert wird.
2. Es werden Templates eingesetzt, welche vom Compiler ausgefüllt werden. Die tatsächlichen Werte werden vom Entwickler im Configurationsbereich der Data App eingetragen.
3. Über die offiziele Java-Bibliothek, welche es erlaubt Entitäten des Infomodells zu erzeugen und zu de-/serialisieren, werden die entsprechenden Objekte bei Bedarf erzeugt und dem Nutzer zurückgegeben.
4. Es wird eine Repräsentation des IDS Infomodells in Nukleus, dem Typsystem von D°, erzeugt und verwendet, um bei Bedarf entsprechende Entitäten zu erzeugen. Diese werden über eine entsprechende Abbildung in das erwartete JSON-Format übertragen und zurückgegeben.

Alle diese Ansätze haben eigene Vor- und Nachteile.

Option 1 ist am einfachsten umzusetzen, verlagert aber die Aufgabe, sowie die damit verbundene Komplexität, eine korrekte JSON-Repräsentation der Metadaten bereitzustellen auf den Entwickler welcher die Applikation umsetzt.

Option 2 ist eine Erweiterung von Option 1 und macht Gebrauch von der Template-Engine, welche bereits Teil des Compilers ist und verwendet wird, um Teile der D°-Applikation zu generieren. Es würde somit keine Änderungen am Technologiestack und den Abhängigkeiten von D° geben. Der Nachteil dieser Lösung ist, dass potentiell mit jeder neuen Version IDS Infomodells neue Templates mit geänderten Feldern bereitgestellt werden müssen.

Option 3 verlagert diese Komplexität in die Java-Bibliothek, welche zur Verfügung steht. Der Entwickler muss nur noch die richtigen Werte für die relevanten Felder bereitstellen, alles weitere wird im Code der Data App umgesetzt. Der Nachteil dieser Lösung ist, dass der gesamte Technologiestack und sämtliche Abhängigkeiten der IDS Infomodell Java-Bibliothek Einzug in die D°-Applikation erhält. Dies kann zu Konflikten und Inkompatibilitäten führen und verursacht eine massive Steigerung in der Größe der erzeugten Applikation.

Option 4 verfügt über die selben Vorteile wie Option 2 ohne die genannten Nachteile aufzuweisen. Der Nachteil dieser Lösung ist, dass ein hoher initialer Aufwand entsteht, um eine saubere Abbildung von Nukleus Entitäten zum IDS Infomodell bereitzustellen. Darüber hinaus entsteht hier potentiell erneut Aufwand mit jeder neuen Version des IDS Infomodells. Dieser Mehraufwand kann dabei von unerheblich bis massiv jede Ausprägung annehmen, abhängig von der jeweiligen Version des Infomodells.

Für D° wird die Umsetzung von Option 2 angestrebt, aber eine temporäre Nutzung von Option 1 in Betracht gezogen, bis die Umsetzung von Option 2 für einen entsprechend stabilen Release des Infomodells abgeschlossen ist.

Swagger UI

Die Bereitstellung einer Swagger UI kann die Verwendung bzw, das Testen einer Applikation erheblich vereinfachen. Diese Funktionalität wird von D°-Applikationen aktuell nicht geboten, ist aber durchaus wünschenswert und befindet sich in Planung.

Das Problem was sich dabei ergibt ist das Folgende: Als Eingabe einer D° Applikation wird ein JSON-Objekt erwartet, welches valide, serialisierte Nukleus-Instanzen enthält. Daraus folgt, das die Eingabe einer D°-Applikation ein String ist, welcher ein JSON-Objekt enthält, dessen Werte selber JSON-Strings sind. Hieraus ergeben sich wenig intuitive Konstrukte, da es notwendig ist die "inneren" JSON-Objekte zu escapen.

Dies kann gegebenenfalls durch die Client Bibliothek, welche für Q2/2021 in diesem Projekt geplant ist, vereinfach oder sogar gelöst werden.

Status

Die Statusinformationen, welche von einer App Store Template App abgefragt werden können, sind nicht näher definiert. Aus diesem Grund und da für D° aktuell kein Bedarf an einer solchen Funktionalität besteht, wird hier keine Änderung an D° vorgenommen.

Config

Das App Store App Template sieht es vor, dass Applikationen von außen zur Laufzeit konfiguriert werden können. Außerdem kann die Konfiguration zur Laufzeit auch abgefragt werden. Dies passt nicht in das Konzept von D°. D° erzeugt im Rahmen des Übersetzungsvorgangs eine Applikation, welche nicht modifiziert/umkonfiguriert werden kann, da dies unter Umständen weitreichende Folgen haben kann, beispielsweise auf die Auswertung von Policies. Aus diesem Grund wird für diese Komponente keine Änderung an D° vorgenommen.

Ein möglicher Workaround (welcher bereits jetzt möglich ist) wäre es, die einzelnen Aktivitäten in der Data App entsprechend mit Parametern auszustatten, welche das Verhalten verändern. Somit würden die entsprechenden Teile der Konfiguration als Teil der Eingabeparameter an die Applikation übergeben. Dies sorgt dafür das die Planung und Implementation der einzelnen Aktivitäten unter Umständen komplexer ist.

Policies

App Store Template Apps erlauben es die verwendeten Policies zur Laufzeit zu setzen. Das mag für einige Anwendungsfälle nützlich sein, widerspricht aber grundlegend dem Konzept von D°. Die dafür relevanten Gründe wurden im vorherigen Abschnitt (Config) aufgezeigt.

Aus diesem Grund kann und wird diese Funktionalität nicht in D° übernommen. Es wäre noch zu diskutieren, ob eine entsprechende lesende Schnittstelle sinnvoll ist (dies gilt auch für das App Store Template), aber dies würde wieder Probleme mit sich bringen. Und zwar müsste die automatische Transformation von IDS Policies in das Format, welches von der Usage Control Lösung in der Applikation verwendet wird, funktionieren. Dies ist aber nicht gegeben. Dabei wird davon ausgegangen, dass die Kommunikation nach außen ausschließlich mit den IDS Policies erfolgt, da ansonsten die Nutzung eingeschränkt/verkompliziert wird.

Generell kann aber von der Bereitstellung einer solchen Schnittstelle abgesehen werden, da die entsprechenden Informationen bereits in den Metadaten der Applikation enthalten sein sollten.