Typen von Data Apps

Versionen

Datum	Notiz	Version
31.03.2020	Erste Veröffentlichung	1.0

Autoren

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

Über generierte Data Apps

Betrachtet man Programmcode, welcher mit D° geschrieben wurde, wird deutlich, dass neben der Konfiguration der Data App nur die auszuführende Logik abgebildet wird. Es wird nicht beschrieben um welche Art von Data App es sich handelt und wie auf diese später Zugegriffen werden soll.

Das nachfolgende Codebeispiel zeigt deutlich, was zuvor beschrieben wurde.

```
App configuration
        application namespace : de_fhg_isst_oe270.degree
        application name : degreeEchoApp
        application version : "0.0.1-1-SNAPSHOT"
        application port : 8080
        startup policies : UseNotBeforeTimeStamp2010, UseNotAfterTimeStamp2020
        JWT signing key : "..."

App code
        [payload = $Text] -> begin
                echoMsg = $Text(@write["Received message:"]);
                UnconstrainedPrintToConsole[echoMsg];

                EchoServicePrintToConsole[payload];

                echoMsg = $Text(@write["End of message."]);
                UnconstrainedPrintToConsole[echoMsg];

                return [payload];
        end
```

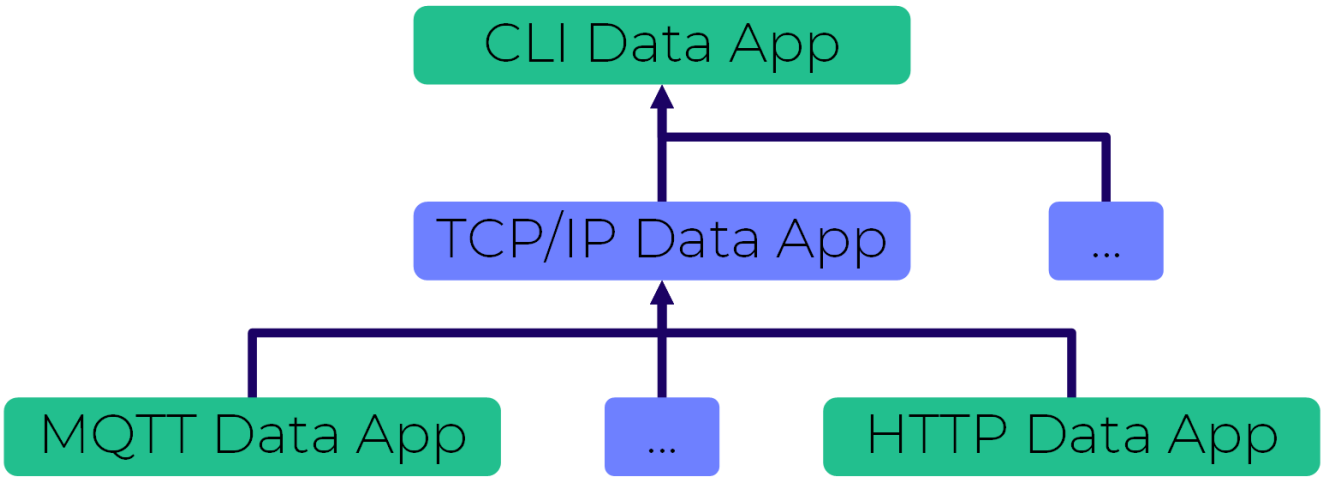
Die frühen Versionen von D° konnten ausschließlich Data Apps generieren welche einen einzelnen HTTP-Entpunkt unter der URL `/process` bereitstellen. Dies hatte im Rahmen der Entwicklung diverse Vorteile, da übersetzte Data Apps einfach getestet werden konnten und Randbedingungen nicht betrachtet werden mussten. Dennoch ist es (langfristig) notwendig, dass D° es ermöglicht weitere Arten von Data Apps zu generieren. Andernfalls sind die sinnvollen Nutzungsszenarien von D° stark eingeschränkt.

Da der aktuelle Entwicklungszustand von D° einen vertikalen Prototypen zur Verfügung stellt, sollen die diversen Systeme nach und nach um zusätzliche Funktionalitäten erweitert werden. Das vorliegende Dokument beschreibt, wie D° verschiedene Arten von Data Apps generieren kann, welche Typen aktuell möglich sind, welche eventuell in einer späteren Version umgesetzt werden und welche Typen Data Apps durch D° eventuell ausgeschlossen sind.

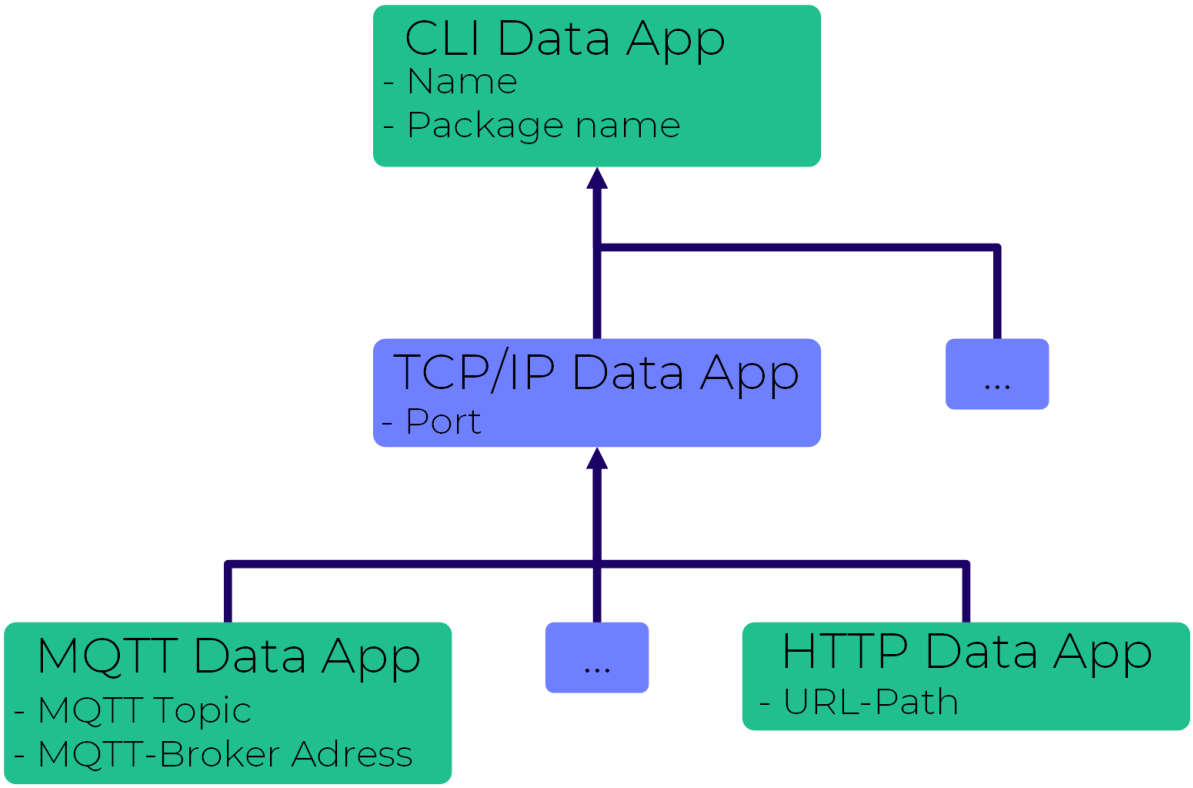
Identifikation von Data App Typen

Wie eingangs erwähnt enthält die Programmlogik keine Informationen darüber, welche Art von Data App generiert werden soll. Diese Informationen müssen demnach aus dem Konfigurationsteil der Data App bezogen werden. Von der Möglichkeit die Art der generierten Data App über Parameter, welche dem Compiler übergeben werden, zu bestimmen wird abgesehen. Dies hat den Grund das Data Apps für gewisse Zwecke und Szenarien entwickelt werden und daher auch der Typ der generierten Data App an das Szenario angepasst wird. Werden die notwendigen Informationen in den Teil des Programmcodes, welcher zur Konfiguration verwendet wird, verlagert, ergibt sich der Vorteil, dass alle Informationen welche zur Übersetzung einer Data App notwendig sind, im entsprechenden Programmcode enthalten sind. Dies ist stärker zu gewichten, als potentielle Vorteile durch die Einstellbarkeit über Compiler-Parameter.

Somit ist die Frage zu beantworten, welche Parametersätze für unterschiedliche Arten von Data Apps notwendig sind. Diese müssen nicht vollständig disjunkt sein. Es ist lediglich notwendig, dass eine eindeutige Zuordnung von Parametersätzen auf Data Apps möglich ist. Es ist möglich die unterschiedlichen Data Apps zu gruppieren und in eine Baumstruktur zu überführen, wodurch eine Taxonomie von Data App Arten entsteht. Die nachfolgende Abbildung enthält eine simple mögliche Taxonomie von Data Apps. Dabei handelt es sich bei den blauen Knoten um abstrakte Data App Typen, welche nicht direkt verwendet werden können. Nur die grün gekennzeichneten Knoten können vom D°-Compiler erzeugt werden. Dabei kann rein aus der Taxonomie nicht abgelesen werden, welche Data App Arten direkt erzeugt werden können.



Reichert man die vorherige Darstellung noch um die zuvor definierten Parametersätze an, zeigt sich welche Parameter notwendig sind, um einen spezifischen Data App Typen zu erzeugen. Dabei wird einfach der Pfad vom zu instanziierenden Typen bis hin zur Wurzel des Baumes verfolgt und alle Parameter die in den Knoten auf dem Pfad vorhanden sind eingesammelt. Die Gesamtheit dieser eingesammelten Parameter ist der spezifische Parametersatz für den jeweiligen Data App Typen. In der nachfolgenden Abbildung wird das vorherige Beispiel um die entsprechenden Parameter erweitert.

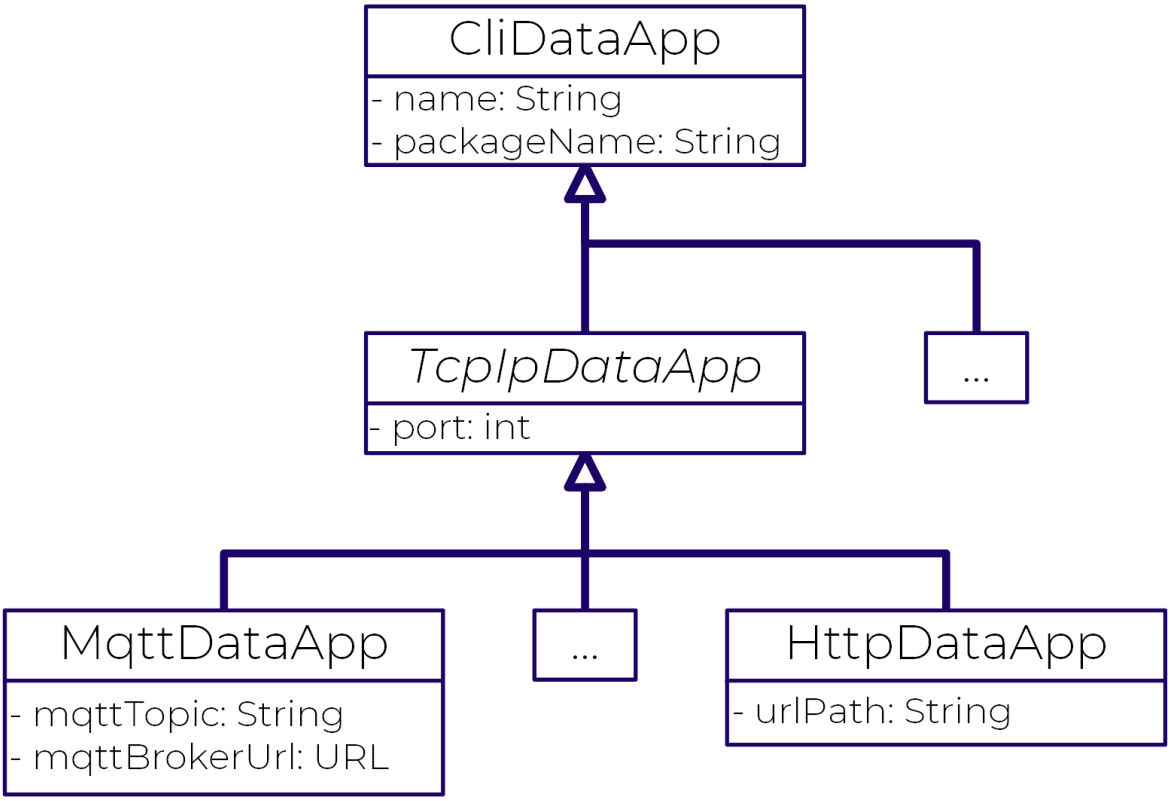


Diese mit Parametern versehenen Taxonomie ist bei sorgfältiger Entwicklung ausreichend genau, um die verschiedenen Data App Typen, welche D° unterstützten soll, zu erzeugen.

Generierung von Data Apps

Zuvor wurde allgemein beschrieben, wie eine Taxonomie von Data App Typen aufgebaut werden kann und welche Werte sie enthalten muss. Aus dieser Taxonomie kann mit geringem Aufwand ein entsprechendes Klassendiagramm und die dazugehörigen Implementierungen erzeugt werden. Das einzige was an dieser Stelle noch fehlt ist der Templatecode für jeden Data App Typen. Dieser Templatecode ist dafür zuständig, dass die entsprechenden Aufrufe (bspw. über MQTT) an die Logik, welche die Applikation abbilden soll, weitergeleitet werden und Ergebnisse und Rückgabewerte über den selben Kanal ausgeliefert werden.

Die nachfolgende Abbildung enthält das Klassendiagramm für die Taxonomie, welche beispielhaft in den vorherigen Abschnitten entwickelt wurde.



Wie der Templatecode, welcher an dieser Stelle noch fehlt, konkret ausgestaltet werden muss, lässt sich nicht allgemein beantworten. Dies liegt daran, dass die verschiedenen Arten von Data Apps unterschiedliche Aktionen durchführen müssen, um die geforderte Funktionalität bereitzustellen. Beispielsweise verursacht eine einfache `CliDataApp`, welche direkt über die Kommandozeile angesprochen wird, wesentlich weniger Overhead als eine `MqttDataApp`, welche sich mit einem MQTT-Broker verbinden und für Topics registriert werden muss. Gleichzeitig stehen diverse Wege zur Implementierung der notwendigen Funktionalität bereit, beispielsweise durch die Verwendung von entsprechenden Bibliotheken.

Dennoch lassen einige Aspekte der unterschiedlichen Implementierungen in Abstrakte Konzepte zusammenfassen. Diese gelten für die unterschiedlichen Implementierungen gleichermaßen, werden aber auf unterschiedliche Art und Weite umgesetzt.

Die `CliDataApp`, welche die Wurzel der Taxonomie darstellt, enthält eine Methode, welche die Logik der Data App enthält. Diese Funktion wird vom D°-Compiler erzeugt und verfügt über die passende Signatur, um Parameter als Eingabe zu erhalten und nach der Ausführung als Ausgabe zurückzugeben. Diese Methode wird von den unterschiedlichen Data Apps aufgerufen, sobald die Logik der Data App ausgeführt werden soll. Beispielsweise wird sie bei der `CliDataApp` als Teil der `main`-Methode ausgeführt,Bei einer `HttpDataApp` dagegen wird sie durch eine Methode, welche Aufrufe an den Http-Endpunkt der Data App entgegennimmt, aufgerufen.

Im Allgemeinen ergibt sich eine Implementierung die einen ähnlichen Aufbau wie das nachfolgende Code-Beispiel hat. Das Beispiel hat aus Gründen der Verständlichkeit an vielen Stellen keinen Programmcode sondern nur beschreibende Kommentare. Das Beispiel ist in der Programmiersprache Java geschrieben.

```
public class ExampleDataApp extends SpecificDataApp /* Choose correct Data App type here */ {

        private ExampleDataApp() {
                super(); // The type specific initialization happens here
        }

        private static dataApp = ExampleDataApp;

    public static void main(String[] args) {
        dataApp = new ExampleDataApp();

                // Additional setup required for the chosen Data App type (e.g. starting a spring context)
    }

        public String executeDataAppLogic(String input) {
                InputScope inputScope = ...; // Input handling
                OutputScope result = process(inputScope); // The process method is generated by the D°-compiler
                String result = result...; // Output handling

                return result;
        }

}
```

Mögliche Data App Typen

An dieser Stelle soll eine Auswahl von möglichen Data App Typen, sowie deren notwendigen Parametern aufgelistet werden. Eine Nennung innerhalb der Liste bedeutet aber nicht zwingend, dass die Implementierung in der nächsten Version von D° stattfindet. Gleichzeitig erhebt die Auflistung keinen Anspruch auf Vollständigkeit. Es werden einfach mehrere Beispiele gegeben, welche sich in D° umsetzten lassen würden.

Dabei wurde TCP als repräsentatives Beispiel für die Transportschicht des OSI-Modells und HTTP sowie MQTT als Repräsentanten für die darüber liegenden Schichten ausgewählt. Die Taxonomie könnte um weitere entsprechende Protokolle und somit Data Apps erweitert werden. Beispiele für solche möglichen Erweiterungen sind Zigbee, CoAP und Bluetooth.

HTTP Data App

Bei der HTTP Data App handelt es sich um den Data App Typen, welcher während der Entwicklung von D° als erstes umgesetzt wurde. HTTP Data Apps können über das Netzwerk bzw. das Internet verfügbar gemacht werden. Durch die große Vielfalt von verfügbaren REST-Clients und entsprechenden Bibliotheken für verschiedene Programmiersprachen ist es sehr einfach möglich solche HTTP Data Apps in Workflows zu integrieren. Darüber hinaus ist durch die verschiedenen Technologien ein einfaches Testen möglich.

Der Parametersatz von HTTP Data Apps enthält nur einen eigenen Eintrag. Die anderen Parameter werden von den Data App Typen, die in der Taxonomie auf einem Pfad von der HTTP Data App zur CLI Data App liegen, geerbt.

Parameter	Ursprung	Beschreibung
Name	CLI Data App	Der Name der Data App. Wird für die Klassenbenennung des generierten Java Codes und Logausgaben verwendet.
Paket Name	CLI Data App	Der Paketname des generierten Java Codes.
Port	TCP/IP Data App	Der Port unter dem die Data App erreichbar ist.
URL Pfad	HTTP Data App	Der Teil der URL, welcher den Pfad beschreibt unter dem die Data App erreichbar ist.

MQTT Data App

Genauso wie die HTTP Data App verwendet die MQTT Data App TCP als Transportprotokoll. MQTT findet im IoT Bereich starke Verbreitung und stellt aus diesem Grund eine interessante Kommunikationsmöglichkeit für D° dar. Durch die Unterstützung von MQTT würde eine Vielzahl von Geräten zur Kommunikation mit Data Apps befähigt.

Parameter	Ursprung	Beschreibung
Name	CLI Data App	Der Name der Data App. Wird für die Klassenbenennung des generierten Java Codes und Logausgaben verwendet.
Paket Name	CLI Data App	Der Paketname des generierten Java Codes.
Port	TCP/IP Data App	Der Port unter dem die Data App erreichbar ist.
MQTT Topic	MQTT Data App	Das Topic das bei der Kommunikation mit dem MQTT Broker verwendet wird.

MQTT Broker	MQTT Data App	Die URL unter dem der MQTT Broker erreichbar ist.

CLI Data App

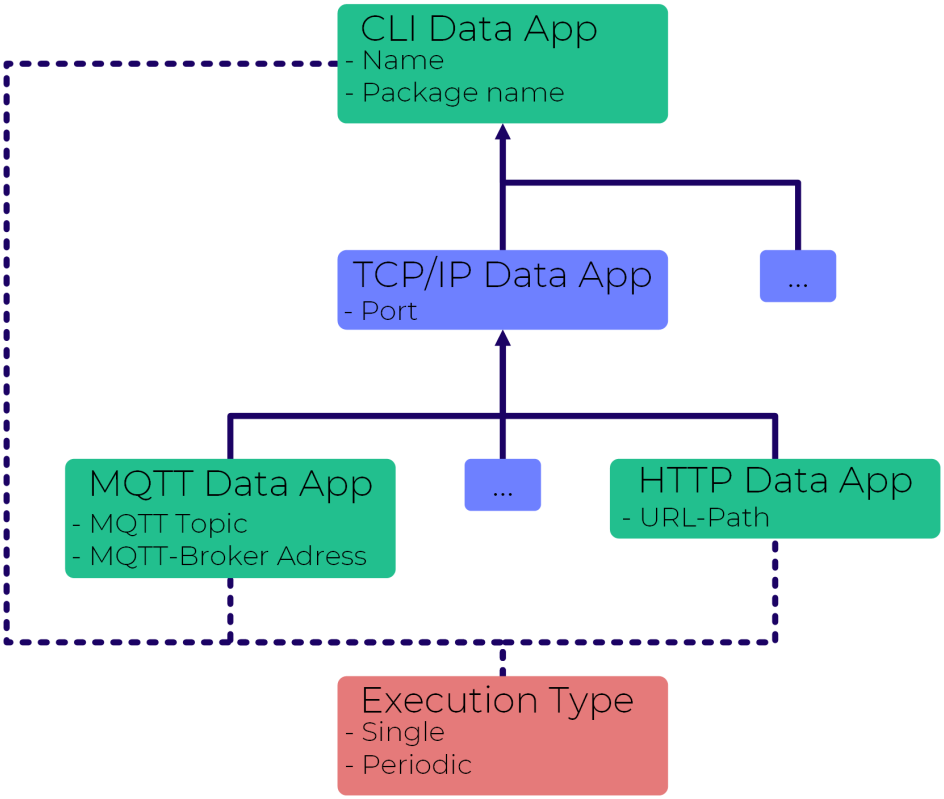
Bei der CLI Data App handelt es sich um die elementarste Form von möglichen Data Apps. Sie unterstützt keine Transportprotokolle und wird direkt über die Kommandozeile angesprochen. Sämtliche anderen Arten von Data Apps basieren auf dieser CLI Data App.

Parameter	Ursprung	Beschreibung
Name	CLI Data App	Der Name der Data App. Wird für die Klassenbenennung des generierten Java Codes und Logausgaben verwendet.
Paket Name	CLI Data App	Der Paketname des generierten Java Codes.

Ausführungsverhalten von Data Apps

Durch das zuvor beschriebene Vorgehen ist beschrieben und definiert, wie neue Typen von Data Apps zu D° hinzugefügt werden können. Dennoch bietet das beschriebene Verfahren nicht die maximal mögliche Flexibilität, da alle Data Apps das selbe innere Verhalten haben: Sie werden aufgerufen (bzw. im Fall der CLI Data App gestartet) und führen einmalig ihre Applikationslogik durch. Hier sind noch andere Arten von Ausführungsverhalten denkbar und wünschenswert. Beispielsweise eine periodische Ausführung oder eine Ausführung in einem gewissen Zeitintervall.

Somit existiert neben dem Data App Typen durch das Ausführungsverhalten eine weitere Dimension in der Beschreibung von Data Apps. Die nachfolgende Abbildung stellt dies dar.



Diese unterschiedlichen Ausführungsverhalten sind für alle Typen von Data Apps gleich. Es ist an dieser Stelle nur zu beachten, dass eine CLI Data App nach der gewünschten Ausführung der Programmlogik terminiert, wogegen eine bspw. HTTP Data App beliebig lange im Leerlauf ausgeführt werden kann und zu jedem Zeitpunkt HTTP-Aufrufe akzeptieren kann. Das heißt es kann durchaus Unterschiede im konkreten Verhalten verschiedener Data App Typen geben, aber dennoch können einzelne Grundmuster identifiziert werden, die gleichermaßen für alle Data App Typen gelten.

Wenn man berücksichtigt, dass jeder Typ von Data App grundlegend über die selben Ausführungsverhalten verfügen kann, macht es wenig Sinn diese Information in der zuvor aufgebauten Taxonomie abzulegen und diese dadurch erheblich zu vergrößern und hierdurch unübersichtlicher zu gestalten. Stattdessen wird eine eigene Struktur entworfen, um die unterschiedlichen Ausführungsverhalten in einer strukturierten Art und Weite zu beschreiben und zu definieren. Es wäre möglich auch für die unterschiedlichen Ausführungstypen eine Taxonomie zu bilden, um eine stärkere Struktur zwischen den Elementen aufzubauen. Da aber im aktuellen Plan nur zwei verschiedene Verhalten betrachtet werden, wird davon aktuell abgesehen. In zukünftigen Versionen wird dieser Schritt möglicherweise noch vollzogen, sofern weitere Verhalten in D° implementiert werden.

Soll nun eine Data App entwickelt werden, ist im ersten Schritt zu bestimmen welchen Typ sie haben soll. Hierdurch wird ein Satz von Parametern vorgegeben, die angegeben werden müssen. Im zweiten Schritt muss das gewünschte Verhalten der Data App definiert werden und das dazu passende Ausführungsverhalten ausgewählt werden. Hierdurch werden eventuell weitere notwendige Parameter vorgegeben. Durch diese zwei Schritte ist es möglich auf einfache Art eine Data App des gewünschten Typs zu entwickeln.

Nachfolgend wird die Aktion, welche die Ausführung der Applikationslogik einer Data App auslöst, als Signal bezeichnet. Der Ursprung des Signals kann dabei unterschiedlicher Natur sein. Bei einer HTTP Data App ist es beispielsweise der Eingang einer passenden HTTP-Anfrage und bei einer CLI Data App der Start der Applikation.

Einzelne Ausführung

Ein eingehendes Signal sorgt für eine einmalige Ausführung der Applikationslogik. Jedes weitere eingehende Signal führt zu einer erneuten Ausführung der Applikationslogik. Dies ist das Standardverhalten für Data Apps und benötigt keine zusätzlichen Parameter, um definiert zu werden.

Periodische Ausführung

Eine anderes Ausführungsverhalten, welches wesentlich vielseitiger einsetzbar ist, ist die periodische Ausführung. Hierbei gibt es zwei Signale: Eines zum Starten der Ausführung und eins zum Beenden. Sobald die Ausführung begonnen wird, wird die Applikationslogik periodisch ausgeführt. Dabei wird der Zeitabstand zwischen zwei Ausführungen durch eine parametrisierte Periode bestimmt. Ist die gewählte Periode kleiner oder gleich der Ausführungszeit der Applikationslogik, läuft die Applikationslogik in einer Endlosschleife.

Parameter	Beschreibung
Periodendauer	Zeitabstand der zwischen zwei konsekutiven Ausführungen der Applikationslogik liegen soll.

Ausgeschlossene Data App Typen

Es sind weitere Typen von Data Apps denkbar, welche bisher nicht nur nicht betrachtet wurde, sondern durch D° ausgeschlossen sind. Dies ist dem Aufbau der Sprache sowie des Compilers geschuldet. Beispielsweise erlaubt D° es nicht eine HTTP Data App mit mehreren (funktional unterschiedlichen) Endpunkten zu definieren. Dies liegt darin begründet, dass eine in D° entwickelte Data App jeweils genau einen Workflow und die dazugehörigen Nutzungsbedingungen abbildet. Hierdurch entsteht zwangsläufig Overhead, wenn durch D° eine HTTP API bereitgestellt werden soll, welche aus verschiedenen Endpunkten besteht, da für jeden Endpunkt eine eigene Data App betrieben werden muss.

Ebenso ist es nicht möglich Data Apps mit integrierter graphischer Nutzeroberfläche zu entwickeln. Es ist lediglich möglich eine separate Oberfläche (in einer anderen Programmiersprache) zu entwickeln, welche auf die Schnittstellen, welche die jeweiligen Data Apps bereitstellen, zugreifen um Inhalte anzuzeigen und Interaktionen zu erlauben.