Das Kitomer-System[1]

Markus Bon
Universität Kaiserslautern
bon@informatik.uni-kl.de

Abstract: Produktentwicklung ist ein langwieriges Geschäft. Eine Chance, die Entwicklungszeiten zu verkürzen, und die Kooperation zwischen beteiligten Partnern zu verbessern, ist die Definition firmenübergreifender Geschäftsprozesse, die von einem geeigneten Workflow-Management-System (WfMS) ausgeführt und überwacht werden können. Das bei uns entwickelte System „Kitomer“ (Kooperation interoperierender, heterogen mehrteiliger Workflows) ist so ein System. Im folgenden Artikel wird der erste Prototyp beschrieben; nach einem kurzen Überblick über die grundlegende Architektur folgt eine mehr technische Beschreibung der Umsetzung.

keywords: Heterogenität, Workflow, Crossorganisational, firmenübergreifend, Kontrollfluß, Prototyp

1. Motivation

Aufgrund steigenden Wettbewerbs („global players“) und immer kürzerer Entwicklungszeiten ist der klassische Weg der Produktentwicklung vom Reißbrett bis zur Produktion nicht mehr gangbar. Oft finden sich die „Entwicklungsteams“ auch nicht mehr an einem Ort zusammen, sie sind vielmehr über viele auf ihre jeweilige Teilaufgabe spezialisierte Firmen weltweit verteilt. Dies erfordert ein hohes Maß an Koordination! Eine Chance, dieses Problem anzugehen, bieten Workflow-Management Systeme. Durch Definition firmenübergreifender Geschäftsprozesse kann die Kooperation zwischen den Partnern deutlich verbessert werden. Probleme bereitet dabei jedoch die Heterogenität der IT-Infrastruktur; da diese in jeder Firma unabhängig „gewachsen“ ist, sind die eingesetzten (WfM-) Systeme alles andere als kompatibel zueinander. Gebraucht wird folglich eine übergeordnete Komponente, die über einen globalen Workflow die lokalen Systeme („Wf-Inseln“) ansteuert, und die Rolle des Boten und Dolmetschers übernimmt. Mit „Kitomer“ (**K**ooperation **i**nteroperierender, he**t**er**o**gen **m**ehrteiliger Wo**r**kflows) wollen wir genau so ein System zur Verfügung stellen.

2. Aufgaben der Koordination

Durch die Verteilung des Gesamtablaufs auf viele lokale Systeme muß für eine entsprechende Koordination des Ablaufs gesorgt werden. Diese umfaßt:

- *Interpretation des globalen Workflow-Schemas;*
  Der globale Geschäftsprozeß, der die Zusammenarbeit zwischen den teilnehmenden Firmen regelt, wird durch einen globalen Workflow beschrieben. Dieser muß in einer Form vorliegen, die dem System ermöglicht, zum richtigen Zeitpunkt den richtigen Teilworkflow auszuwählen und mit notwendigen Daten zu versorgen.

- *Initiieren der erforderlichen Workflow-Exemplare auf den Inseln;*

1. „Kitomer“ ist der Name eines Planeten aus Roddenberries Star Trek Universum. Dort fand 2293 eine Friedenskonferenz statt, an der viele (ganz heterogene) Völker teilnahmen.

Jede Firma besitzt lokale Geschäftsprozesse, die ihren Teil zur Durchführung der Gesamtaufgabe beschreiben. Sie liegen in Form von Workflow-Schemata vor, die von dem lokal eingesetzten WfMS verarbeitet werden können. Sobald es der globale Workflow erfordert, werden davon (lokal) Instanzen gebildet, die erforderlichen Parameter gesetzt und die Ausführung durch eine lokale Wf-Ausführungseinheit („Engine“) gestartet.

- *Kontrolle der Abhängigkeiten zwischen den lokalen Workflows;*
  Da die Grundaufgabe des globalen Workflows der Unterstützung von Kooperation zwischen den Inseln dient, kann die Ausführung der lokalen Workflow-Instanzen nur in den seltensten Fällen vollkommen autonom geschehen. Vielmehr bestehen mehr oder weniger komplexe Abhängigkeiten zwischen den Inseln. Die WfMC identifiziert beispielsweise drei Grundtypen von Abhängigkeiten: Anordnung (CHAIN), Auftrag (NEST) und Synchronisation (SYNC). Diese Abhängigkeiten müssen überwacht und die zugehörigen Wf-Instanzen lokalisiert werden. Weiterhin muß die Möglichkeit der Kommunikation zwischen den Inseln angeboten werden.

- *Datenfluß*
  Eine wichtige Rolle bei der Kooperation besteht auch darin, gemeinsam genutzte Daten auszutauschen, bzw. abzugleichen. Da diese Daten von verschiedenen Inseln stammen, kann eine Konvertierung notwendig werden, es können auch verschiedene (durchaus gültige) Versionen existieren, und Sicherheitsaspekte dürfen natürlich auch nicht außer Acht gelassen werden.

- *Unterstützung von „Monitoring“ und weiterer Administrationsfunktionalität;*
  „Monitoring“ ermöglicht das Überwachen eines Ablaufs zum Ausführungszeitpunkt. Zu diesem Zweck kann durch Abfragen der lokalen Inselzustände ein Gesamtzustand für den globalen Workflow abgeleitet werden. Auf diese Weise läßt sich der Fortschritt des laufenden Prozesses überwachen. Falls ein Eingriff von qualifizierter Seite notwendig wird (Fehlerfall), soll dies durch geeignete Werkzeuge ermöglicht werden.

- *Protokollierung des Ablaufs („History“).*
  Zu einer nachträglichen Analyse werden Informationen mitprotokolliert werden, um die Durchführung des Workflows nachvollziehbar zu machen. Diese können dann beispielsweise zur Fehlersuche genutzt werden, aber auch zur Beurteilung der Qualität der Durchführung. Bei der Optimierung eines Prozesses sind History-Daten von unschätzbarem Wert.

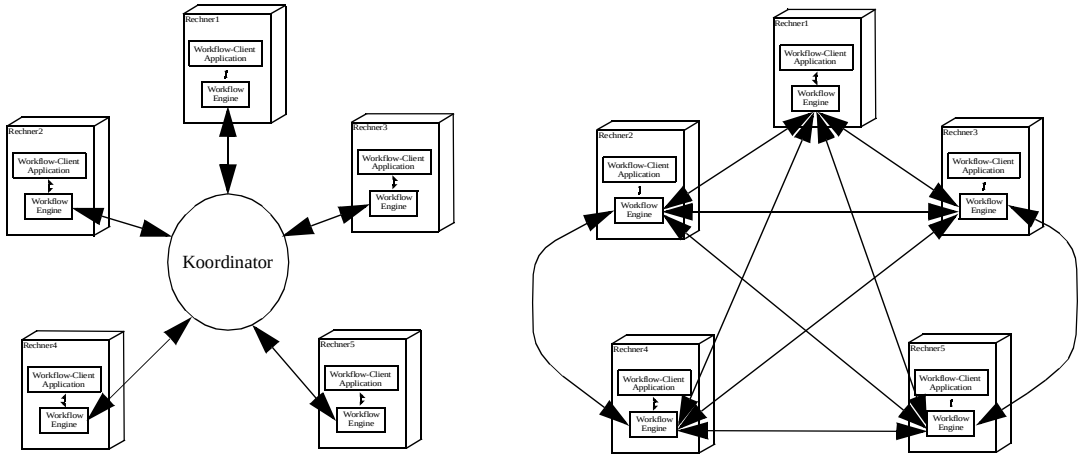
3. Architekturansätze

Zur Durchführung der beschriebenen Aufgaben sind verschiedene Ansätze denkbar. Zwei Extremformen davon werden im folgenden Abschnitt vorgestellt. Im zentralen Fall übernimmt eine spezielle Komponente die Koordination, im verteilten Fall wird diese Aufgabe von den Inseln selbst übernommen.

3.1 Zentrale Kontrolle

Auf der Koordinatorebene ist eine vollständige Beschreibung des globalen Workflows sichtbar. Zur Kontrolle der Ausführung dient eine zentrale Komponente, die dieses globale Schema kennt. Diese Komponente bezeichnen wir als Koordinator. Dem Koordinator ist nicht nur die Beschreibung des Gesamtprozesses zugänglich, auch die Zielsysteme, auf denen die entsprechenden Sub-Workflows ausgeführt werden sollen, sind ihm bekannt. Weiterhin läuft auch die

Kommunikation zwischen den Inseln über den Koordinator. Aufgrund dieser vollständigen Information sind die zuvor geforderten Aufgaben recht einfach lösbar, insbesondere die Überwachung des Gesamtzustands und das Mitprotokollieren des Ablaufs.
Die Abwicklung der Kommunikation über den Koordinator stellt Anforderungen sowohl an den Koordinator selbst, als auch an die Inseln: Der Koordinator muß den Inseln eine API anbieten, die das Versenden von Nachrichten ermöglicht. Durch die Kenntnis des globalen Schemas ist es ihm möglich, selbstständig die Zielinsel festzustellen und die Nachricht auszuliefern. Weiterhin kann eine so verschickte Nachricht protokolliert werden.
Auf der Inselseite benötigen wir zur Zusammenarbeit mit dem Koordinator eine API, die ihm die Auslieferung einer Nachricht an die Insel gestattet. Darüber hinaus muß die Insel-API das Erzeugen und Initiieren von Workflows, sowie die Abfrage des lokalen Zustands durch den Koordinator unterstützen.



a) zentrale Kontrolle **b) voll verteilt - dezentrale Kontrolle**

Abbildung 1 Kontrolle der Ausführung

„Zentrale Komponente“ bedeutet „logisch zentral“, es muß sich dabei mitnichten um ein monolithisches System handeln. Der Koordinator an sich kann durchaus verteilt umgesetzt werden.

3.2 Dezentrale Kontrolle

Eine andere Idee besteht darin, den Gesamt-Workflow verteilt zu realisieren (Abbildung 1b). Der jeweils aktive Knoten trifft die Entscheidung, wie weiter zu verfahren ist, und gibt die Kontrolle entsprechend weiter. Hierbei muß jedoch nicht nur ein neuer Workflow auf der Insel erzeugt werden, vielmehr müssen auch alle für den weiteren Ablauf nötigen Informationen weitergegeben werden. Es ergeben sich jedoch nicht zu unterschätzende Probleme: Jedes auf einer Insel befindliche WfMS muß diese Art der Kommunikation von Workflow-Zuständen unterstützen, was eine gravierende Erweiterung der bestehenden Systeme erfordert. Das Lokalisieren der Insel, die als nächste die Kontrolle übernimmt, ist ebenfalls problematisch: durch das Fehlen einer zentralen Instanz ist der Einsatz teurer Mechanismen wie „Broadcast“ notwendig. Das Feststellen eines globalen Zustands und „Monitoring“ wird durch die Migration von Insel zu Insel, bzw. der gleichzeitigen Aktivierung mehrerer Inseln bei Verzweigung des Ablaufs, geradezu unmöglich. Auch die Wünsche, eine Art „Historie“ in Form eines Protokolls zu führen, bzw. im Fehlerfall in einem früheren Schritt wieder aufzusetzen, sind durch das Fehlen einer zentralen Überwachung nahezu unerfüllbar. Eine Insel beendet nach Erfüllen ihres Teilablaufs ihre Aktivität mit Weitergabe der Kontrolle (ansonsten wird ein aufwendiges Beenden der ver-

teilten Workflows notwendig). Insbesondere steht sie dann auch nicht mehr zur Auskunft über den lokalen Zustand oder die bisherige Vorgeschichte zur Verfügung.

4. Architektur von Kitomer

Für Kitomer haben wir, aus den geschilderten Gründen, einen zentralen Ansatz gewählt. In Abbildung 2 sind die Komponenten dargestellt. Bei „The Tool“ handelt es sich um die Definitionszeitunterstützung, die sowohl beim Bearbeiten der globalen Workflow-Schemata, als auch bei der Anpassung der lokalen Insel-Workflows Unterstützung bieten soll. Mit ihrer Hilfe werden die Informationen in den Datenspeichern „Globale Schemata“ (**GS**), „Struktur“ (**S**) und „Registry“ (**R**) gepflegt. In GS werden Informationen zu den globalen Schemata gesammelt. In unserem ersten Ansatz haben wir hier eine einfache Kontrollflußbeschreibung gewählt, die den globalen Prozeß beschreibt. Es sind aber auch weniger explizite Beschreibungsformen denkbar. In **R** werden die lokalen Workflows verwaltet, es finden sich hier beispielsweise Informationen zum Lokalisieren der zugehörigen Insel oder den zum Initiieren notwendigen Parametern. In **S** werden die Abhängigkeiten zwischen den lokalen Workflows beschrieben.

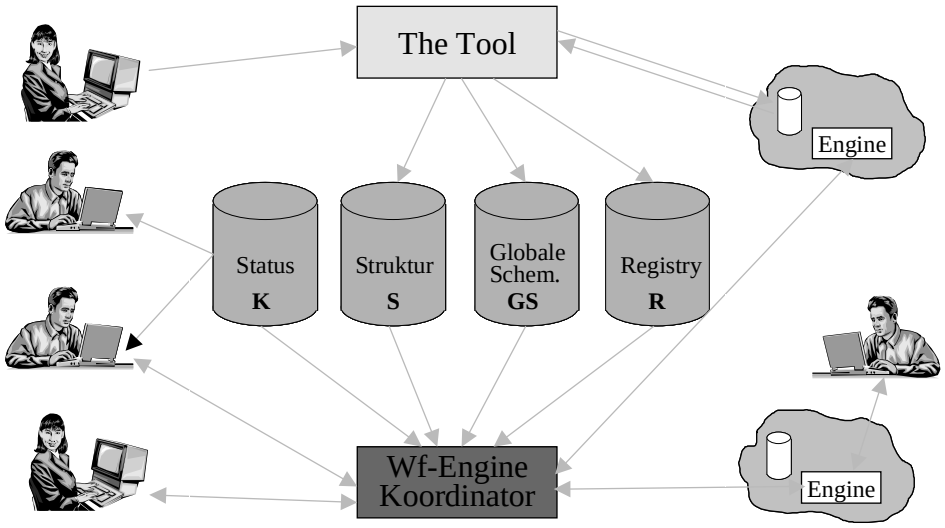


Abbildung 2 Die komponenten von Kitomer

Der Koordinator ist die zentrale Komponente der Laufzeitumgebung. Über ihn können globale Workflows instanziert und gestartet werden, das Verhalten des Gesamtsystems überwacht und der Status bestimmter Wf-Instanzen beobachtet werden. Beim Erzeugen einer neuen Instanz wird für jeden Wf eine eigene Wf-Engine erzeugt, die nur für die Ausführung dieser speziellen Instanz zuständig ist.

5. Der Prototyp

Bei der ersten prototypischen Implementierung wurden noch sehr viele Einschränkungen vorgenommen. Die Wf-Engines zur Ausführung der globalen Workflows sind noch geradezu primitiv, die Nachrichtenformate proprietär, und die Insel-funktionalität wird durch eine einfache Worklist nachgebildet. Trotzdem war es eine interessante Erfahrung, ein erstes Gefühl für die durch die Verteilung der Subworkflows aufkommenden Probleme zu entwickeln. Im nun folgenden Teil wollen wir die Umsetzung der Komponenten im Prototyp näher vorstellen. In Ab-

bildung 3 ist das Zusammenspiel der Komponenten skizziert. Eine zentrale Rolle spielt dabei der Nachrichtenserver (*MSGQ*), der eine asynchrone Kommunikation der Komponenten untereinander ermöglicht. Über eine *Koordinator-GUI* können globale Workflows gestartet werden, zu deren Ausführung jeweils eine eigene *Engine* erzeugt wird. Alle zur Durchführung notwenigen (statischen) Informationen können über die *Repository*-Komponente erfragt werden, der aktuelle Zustand einer Workflow-Instanz wird von *Control* verwaltet.

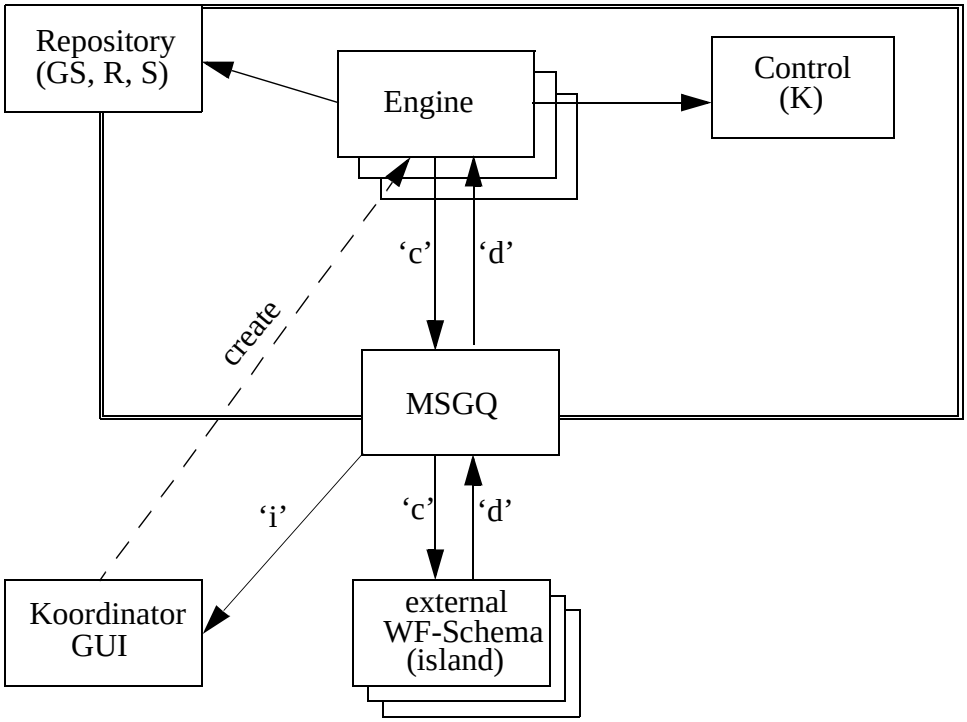


Abbildung 3 Zusammenspiel der Komponenten (Laufzeit)

5.1 Die Koordinator-GUI

Der Koordinator ist die zentrale Komponente bei der Ausführung eines globalen Workflows. Die GUI gliedert sich grob in drei Teile: Workflow, Zentrale und Protokoll. Im Bereich Workflow (Abbildung 4) werden alle definierten globalen Schemata angeboten. Nach Auswahl eines Schemas wird dieses im Sichtbereich dargestellt und kann anschließend instanziert werden. Die „Definitionssprache“ ist dabei sehr einfach gehalten. Durch ‘,’ getrennte Sub-Workflows werden parallel ausgeführt, die Zeilen sequentiell abgearbeitet. Kontrollstrukturen sind noch keine vorgesehen. Das Ausführungsmodul kann aber beliebig reimplementiert/erweitert und sehr einfach ausgetauscht werden.

Abbildung 4 Auswahl eines globalen Workflows

Die Auswahl „Zentrale“ (Abbildung 5) dient dem Überwachen aller sich in Ausführung befindlicher Workflow-Instanzen. In tabellarischer Form wird jeweils die ID der Instanz, der aktuelle Status (initiated, running, active, suspended), der Typ des Workflows (über den Schemanamen) und der aktuell ausgeführte Schritt angezeigt.

Abbildung 5 einfaches Monitoring der in Ausführung befindlichen Workflows

Weiterhin besteht die Möglichkeit, sich ein Bild über das Verhalten des Gesamtsystems zu machen. Alle Komponenten (Engines, Inseln) erstatten dem Koordinator in einfacher Form Bericht, beispielsweise über das erfolgreiche Starten einer Engine, das Beenden eines Sub-Workflows oder auch das erfolgreiche Ende des Gesamtablaufs. Diese Ereignisse können mit der Protokoll-Auswahl (Abbildung 6) angezeigt werden.

Abbildung 6 Protokoll des Systemververhaltens

5.2 Insel-GUI

Zu Testzwecken wurden einfache „Inselsysteme“ integriert. Diese bieten die Möglichkeit, sich alle an die Insel gesendeten Aufträge in einer einfachen Worklist anzeigen zu lassen (Abbildung 7), und deren erfolgreiche Ausführung zu quittieren. Der Koordinator wird anschließend davon in Kenntnis gesetzt.

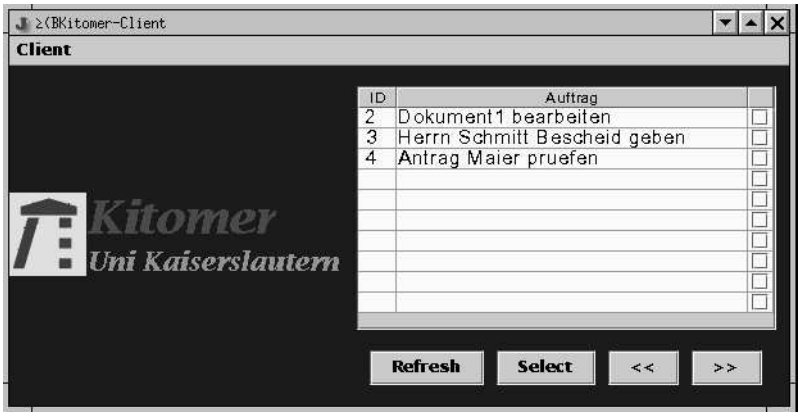


Abbildung 7 Insel-GUI

5.3 Engines

Für jede erzeugte Instanz eines globalen Workflows wird eine neue Engine in einem eigenen Thread erzeugt. Diese überprüft in regelmäßigen Abständen, ob alle momentan auszuführenden

Teilschritte beendet wurden, welche weiteren Schritte zu initiieren sind, oder ob der Workflow bereits erfolgreich abgearbeitet wurde. Die momentane Implementierung besitzt nur einen sehr einfachen Parser zur Interpretation des globalen Schemas, doch läßt sich diese Komponente durch die modulare Struktur des Systems sehr leicht erweitern bzw. auswechseln.

5.4 Repository

Die Repository-Komponente ist die Anlaufstellefür alle Fragen, die das Schema betreffen. Es lassen sich benötigte Informationen über globale Workflows (beispielsweise die Namen der zur Verfügung stehenden Schemata) abfragen und verwalten, weiterhin stehen Informationen zum Lokalisieren der Inselworkflows zur Verfügung.

5.5 Control

Jede Workflow-Instanz ist über eine eindeutige ID im System bekannt. In *Control* werden die zugehörigen Zustandsinformationen verwaltet. In unserem Fall handelt es sich dabei hauptsächlich um den aktuellen Bearbeitungsfortschritt und den zugewiesenen Status (initiated, running, suspended, active, terminated, completed). Momentan erfüllt *Control* somit auch eine gewisse History-Funktion, da auch Einträge beendeter Instanzen erhalten bleiben. Aufgrund der persistenten Speicherung der Zustandsinformation wird auch ein rudimentäres Neustarten nicht vollständig durchgeführter Instanzen ermöglicht.

5.6 MSGQ

Die Kommunikation zwischen den Komponenten findet asynchron statt. Dem Nachrichtenserver *MSGQ* kommt daher eine zentrale Bedeutung zu. Durch ihn findet eine weitestgehende Entkopplung der Komponenten statt, weiterhin wird das System durch die persistente Zwischenspeicherung von Nachrichten robuster gegenüber Systemausfällen. Eine Nachricht besteht aus einer Nachrichtennummer, einem Empfänger (dem Namen des Zielsystems“, einem Nachrichtenrumpf und einer Typangabe. Im System existieren drei verschiedene Arten von Nachrichten: Aufträge (‘c’), Durchführungsbestätigungen (‘d’) und Informationen (‘i’). Aufträge werden dabei von einer *Engine* an die Zielinsel geschickt, diese antwortet nach Erfüllen des Auftrags mit einer Durchführungsbestätigung. Informationsnachrichten können von den Komponenten an den Koordinator geschickt werden, um eine Übersicht über die Geschehnisse im System zu ermöglichen (*Abbildung 6).*

6. DB-Schema

Nachfolgend finden sich die DDL-Befehle, mit denen die zum Betrieb des Prototyps notwendigen Tabellen erzeugt werden können.

```
-- MSG-QUEUE
------------

-- msgID:  fortlaufende, eindeutige Nummer
-- target: Name des Zielsystems
-- body:   Anweisungen an Zielsystem
-- status: n = "new", r = "read"
-- type:   c = "command", d = "done"
```

```
CREATE Table KIT_messageQueue(
msgID   int PRIMARY KEY,
target  varchar(64) NOT NULL,
body    varchar(256),
status  char NOT NULL   CHECK ((status = 'n') or (status = 'r')),
type    char NOT NULL   CHECK ( (type   = 'c')
                    or      (type   = 'd')
                    or      (type   = 'i')
        )
);


-- REPOSITORY
-------------

-- wfID: laufende Nummer als eindeutiger Schlüssel
-- name: Name des Workflows
-- info: kurze Beschreibung

CREATE Table KIT_global_workflow(
wfID    int     PRIMARY KEY,
name    varchar(20),
info    varchar(256)
);
-----------------------------------------------------------

-- wfID: referenziert ID des globalen Workflows
-- step: Nummer des Schritts
-- command: In diesem Schritt auszuführende KF-Anweisung

CREATE Table KIT_global_workflow_step(
wfID    int,
step    int,
command varchar(256),
PRIMARY KEY(wfID, step)
);
-----------------------------------------------------------

-- wfID: laufende Nummer als global eindeutiger Schlüssel
-- name: Name des lokalen Workflows
-- info: kurze Beschreibung
-- target: Name der Insel, die diesen Workflow ausführen kann

CREATE Table KIT_local_workflow(
wfID    int             Primary key,
wfName  varchar(32),
info    varchar(256),
target  varchar(64)     NOT NULL
);
-----------------------------------------------------------
```

```
-- name:  Name der Insel
-- info:  Beschreibung der Insel

CREATE Table KIT_targets(
name    varChar(64)     Primary Key,
info    varchar(256)
);
-------------------------------------------------------------


-- STATUSTABELLE
----------------

-- instId: Identifikator der zugehörigen Engine des Koordinators
-- wfType: ID des instanziierten globalen Workflows
-- aktStep:In welchem Schritt befindet sich die Instanz
-- maxStep:Wieviele Schritte gibt es insgesamt (aus Performancegründen
--         materialisiert)
-- status: Status der Instanz (initiated, running, suspended,
--         active, terminated, canceled)

CREATE TABLE kit_instance_status(
instId  int             Primary Key,
wfType  int,
aktStep int,
maxStep int,
status  char            CHECK ( (status   = 'i')
                    or      (status   = 'r')
                    or      (status   = 's')
                    or      (status   = 'a')
                    or      (status   = 't')
                    or      (status   = 'c')
                    )
);
-------------------------------------------------------------

-- instId: Identifikator der zugehörigen Engine des Koordinators
-- msgId:  Nachricht, auf deren Abarbeitung momentan gewartet wird

CREATE TABLE kit_instance_toDo(
instId  int,
msgID   int,
PRIMARY KEY(instId, msgId));
-------------------------------------------------------------
```

```
-- HILFSTABELLE

--instID: höchste schon vergebene Instanznummer
--msgId:  höchste schon vergebene Nachrichtennummer

CREATE Table KIT_IDs(
instID  int,
msgID   int
);

insert into kit_ids
values (0,0);
```