Abschlussbericht

Auftrag 1201374

22.05.2012

INHALT

ABSCHLUSSBERICHT

22.05.2012

1 GEGENSTAND

In dem Beratungsvertrag vom 02.04.2012 wird die zu erbringende Leistung folgendermaßen beschrieben:

> *Beratungsleistungen zur Erstellung einer KNIME-basierten Softwaremodul-Bibliothek zur Durchführung von Modellrechnungen im Bereich der Lebensmittel-Prozessketten.*

In §4 "Leistungen des Arbeitnehmers" des Beratungsvertrags werden in Punkt (2) "Arbeitsschritte" die Bestandteile einer Lebensmittel-Prozesskette aufgelistet.

1.1 LEISTUNGSBESCHREIBUNG

1.1.1 GENERISCHE DATENSTRUKTUR FÜR ALLE PROZESSINFORMATIONEN (DIESE DATENSTRUKTUR SOLL DURCH ALLE PROZESS-KNOTEN „FLIEßEN“)

Die Modellierung von Lebensmittelherstellungsprozessen macht es notwendig, dass Informationen über Zutaten, Zwischenprodukte, Prozessparameter und Kontaminanten in einer einheitliche Datenstruktur abgelegt werden müssen, um prozessrelevante Informationen (deren Veränderung und „Prozesshistorie“) von einem Prozess-Knoten zum nächsten Prozess-Knoten „weiterzureichen“.

Die Datenstruktur muss mindestens die folgenden Informationen abbilden:

- Zeit t [h]
- Temperatur T[°C]
- pH-Wert
- Druck
- Wasseraktivität a_W
- Konzentration einer Kontaminanten (inkl. Einheit)
- Volumen / Menge aller Materialkomponenten (inkl. Einheit)

Das Volumen der Materialkomponenten wird zu Prozessbeginn und ggf. im Prozessverlauf durch einen Zutatenknoten definiert und ändert sich nur, wenn Kombinationen bzw. Separationen des Materialflusses erfolgen. Die Konzentration einer Kontaminanten wird durch den Kontaminationsknoten initial festgelegt

und vom Berechnungsknoten durch Anwendung entsprechender mathematischer Modelle unter Berücksichtigung relevanter Prozessparameter verändert.

1.1.2 PROZESS-KNOTEN

Der Prozess-Knoten repräsentiert einen Einzelprozess. Dieser Prozess soll charakterisiert sein durch mindestens

- Dauer
- Temperatur (ggf. zeitabhängig)
- pH-Wert (ggf. zeitabhängig)
- Wasseraktivität (ggf. zeitabhängig)
- Druck(ggf. zeitabhängig)
- Mengen und Bezeichnung der Materialien an den Ausgängen

Der Knoten hat mindestens vier optionale Ein- und mindestens vier optionale Ausgänge. Die Eingänge erwarten eine wie in Punkt 1 beschriebene Datenstruktur. Der Prozess-Knoten hat zudem einen weiteren optionalen Eingang für Kontaminanten (siehe Kontaminations-Knoten).

Die Ausgangstabellen gehen aus den Eingangstabellen in folgenden Schritten hervor:

- mathematische „Verarbeitung“ der Materialflüsse inkl. Volumina („Zutaten“ definiert durch Eingänge; „Produkte“ definiert durch festgelegten (bzw. vom Nutzer konfigurierten) Gewichtungsfaktoren) und Berechnung des bzw. Abgleich mit dem Prozess-Gesamtvolumen(s)
- mathematische „Verarbeitung“ der zeitabhängigen Prozessparameter
- entsprechende Aktualisierung der Information in den an die Ausgänge übergebenen Datenstrukturen

1.1.3 KONTAMINATIONS-KNOTEN

Ein Kontaminationsknoten soll die Möglichkeit bieten, im Konfigurationsdialog eine oder mehrere Kontaminanten als Freitext und die jeweilige Menge/Volumen der Kontaminanten als Double-Wert inkl. Einheit einzugeben.

Der Knoten besitzt genau einen Ausgang (keine Eingänge). Der Kontaminationsknoten ist dem Zutaten-Knoten sehr ähnlich. Der Unterschied besteht darin, dass hier Kontaminanten definiert werden, die im Workflow anders behandelt werden als Zutaten. Kontaminations-Knoten münden in Prozess-Knoten.

1.1.4 ZUTATEN-KNOTEN

Der Zutaten-Knoten besitzt keinen Eingang sondern lediglich einen Ausgang. Der Nutzer kann das

Stoffvolumen / Menge einer oder mehrerer (Lebensmittel-) Zutat(en) angeben. Berechnungen werden von dem Knoten nicht durchgeführt.

Zutaten-Knoten münden in Prozess-Knoten.

1.1.5 BERECHNUNGS-KNOTEN

Berechnungs-Knoten sollen die Ausführung mathematischer Berechnungen ermöglichen. Diese Berechnungen ermöglichen die Modellierung der Veränderungen von Konzentrationen / Gehalten an Kontaminanten als auch der Veränderung von Lebensmittelmatrix-Eigenschaften (z.B. pH, a_W) in den jeweiligen Prozessschritten.

Der Berechnungs-Knoten hat zwei Eingänge und einen Ausgang. Ein Eingang wird von einem Datenfluss aus einem Prozess-Knoten belegt; der zweite Eingang wird mit Informationen über die mathematischen Modelle belegt, die für die Berechnungen notwendig sind. Die Informationen über die Modelle kommen aus der Datenbank des BfR.

Der Nutzer wählt ein Berechnungsmodell in der BfR-DB z.B. anhand folgender Suchkriterien aus:

- Agens
- Matrix
- Temperatur
- pH
- Druck
- Wasseraktivität a_W

Der Nutzer muss zudem auch die Möglichkeit haben, ein eigenes Modell (=Funktionsgleichung) mit feststehenden Parametern in den Berechnungs-Knoten einzugeben.

Der Berechnungsknoten führt die geforderte mathematische Berechnung für das Zeitintervall des am Eingang anliegenden Prozesses-Knotens aus und übergibt die Berechnungsergebnisse an den Knotenausgang.

1.1.6 REPORT-KNOTEN

Die Berechnungs- und Prozessdaten müssen einem Report-Knoten übergeben werden können, der eine schriftliche Protokollierung und graphische Aufbereitung sowie den Export dieser Dokumente in eine Datei ermöglicht. Der Report-Knoten muss es insbesondere ermöglichen, dass der Verlauf aller Prozessparameter und die Veränderung der Kontaminante(n)konzentration in Abhängigkeit von der Prozessdauer dargestellt werden kann.

Der Reportknoten erhält als Input eine wie in Punkt 1 beschriebene Tabelle. Er hat keinen (logischen)

Ausgang.

1.1.7 AUTOMATISCHE ERSTELLUNG VON KNIME-WORKFLOWS

In der BfR-Datenbank sind bereits Informationen zu Lebensmittelherstellungs prozessketten hinterlegt, die in KNIME als Workflow automatisch dargestellt werden sollen. Unabhängig von der BfR-Datenbank soll es zudem möglich sein, mit Hilfe von KNIME neue Prozessketten zu definieren und in einem KNIME-Projekt, in einem definierten xml-Schema oder in die BfR-Datenbank abzuspeichern. Ein xml-Schema, dass alle notwendigen Informationen beinhaltet, wurde bereits definiert.

1.2 PRIORISIERUNG

In Abstimmung mit dem BfR wurden zunächst die zu leistenden Übergabepunkte priorisiert. Top Priorität hatte die Erstellung einer generischen Datenstruktur, die alle Aspekte aller oben gelisteten Knoten detailgenau widerspiegelt und gleichzeitig die Mächtigkeit besitzt die komplette Prozesskette inklusive aller Informationen als xml-Datei in einem eigenen Format abzuspeichern. Gleichzeitig muss sie als Übergabepunkt für einen Reportknoten alle berechneten und vom Benutzer definierten Parameter übergeben können. Für diese neu zu schaffende Datenstruktur sollen weitere Knoten entwickelt werden, die den Austausch mit den xml-Knoten in KNIME realisiert. Weiterhin soll es einen Knoten geben, der die in der neuen Datenstruktur zu hinterlegenden Berechnungen in eine Standard-KNIME-Tabellenstruktur überführt, die Informationen über alle berechneten Parameter beinhaltet.

Neben der Datenstruktur wurde die Unterstützung bei der Entwicklung des Prozess-Knotens und die automatische Erstellung von KNIME Workflows hoch priorisiert.

Die Kontaminations- und die Zutatenknoten wurden hintan gestellt. Testdaten sollen aber erzeugt werden können aus der neuen xml-Struktur heraus. Berechnungs- und Reportknoten wurden ebenso hintan gestellt, es sollte aber darauf geachtet werden, dass bei der Definition der neuen xml-Struktur alle Informationen und Berechnungen der Prozesskette für den Reportknoten verfügbar sind.

Der Prozess-Knoten ist zentraler Bestandteil einer Prozesskette und zugleich der Knoten mit der höchsten Komplexität. Aus der Definition der Funktionalität des Prozess-Knotens ergeben sich die Elemente die in der Datenstruktur abzubilden sind. Ist die Datenstruktur fähig, Ergebnisse des Prozess-Knotens zu transportieren und alle Informationen für die Konfiguration des Prozess-Knotens bereitzustellen, dann kann sie auch auf für die Knoten mit geringere Komplexität angewendet werden. Zutaten-, Kontaminations- und Report-Knoten verwendet Teile der Datenstruktur die aus der Definition des Prozess-Knotens folgt. Das unterstreicht die herausragende Rolle des Prozess-Knotens für den erfolgreichen Abschluss des Gesamtprojekts.

2 ARBEITSSCHRITTE

2.1 GENERISCHE DATENSTRUKTUR FÜR ALLE PROZESSINFORMATIONEN

Die Datenstruktur soll alle Prozessbeteiligte - Matrices und Agenzien und deren Mischungen - sowie alle

Prozessparameter – Temperatur, pH, aw, Druck - und deren zeitlichen Verlauf aufnehmen können. In den Zutaten- und Kontaminationsknoten wird die Datenstruktur mit entsprechenden Informationen über Matrices und Agenzien gefüllt. Prozessknoten fügen zur Datenstruktur Prozessgrößen und Daten über den Prozessverlauf hinzu. Weiterhin kann in einem separaten Berechnungsknoten die Konzentrationsveränderungen der Agenzien berechnet werden, optional kann das auch im Prozessknoten realisiert werden – in beiden Fällen wird die Datenstruktur mit den aktuellen Werten aktualisiert. Die Datenstruktur erfasst auch genau welcher Inport mit welchem Outport aus welchem Prozessknoten verknüpft ist. Somit bildet die Datenstruktur eine vollständige Dokumentation der Prozesskette. Ziel ist es, dass Informationen alleine aus dieser Datenstruktur ausreichen um den Zustand und Prozessverlauf im Report-Knoten darstellen zu können.

Für die Realisierung der Datenstruktur wird der Industriestandard XML[1] verwendet. XML bietet durch seinen hierarchischen Aufbau und Erweiterbarkeit die notwendige Elemente, um komplexe Datenstrukturen abzubilden. Zusätzlich wird KNIME mit einer Reihe von Knoten zum Lesen, Schreiben und Verarbeiten von XML Dokumenten ausgeliefert.

Im Rahmen der Beratungsleistung wurde ein XML Schema[2] erstellt. Ein XML Schema definiert alle Elemente eines XML Dokuments, deren Hierarchie sowie Name und Typ der Elementattribute. Das XML Schema dokumentiert nicht nur die Struktur, es kann auch verwendet werden um die Struktur eines XML Dokuments zu verifizieren, d.h. ob alle als "required" gekennzeichneten Elemente und Attribute vorhanden sind und ob die Hierarchievorgabe eingehalten wird.

Die XML Schema Datei pcml-1-0.xsd[3] definiert die Struktur eines XML Dokumentes mit dem Wurzelelement PCML (Process Chain Markup Language). Kindelemente von PCML sind Header, ProcessChain und ProcessChainData. Header beinhaltet Meta-Informationen wie einen Copyright Notiz und eine Beschreibung des PCML Dokumentes und einen Zeitstempel.

Die ProcessChain besteht aus einer Reihe von ProcessNode-Elementen in denen die In- und Outports der Prozess-Knoten sowie physikalische Eigenschaften des Prozess-Knotens definiert werden können.

Mit Java werden eine Vielzahl von Klassen und Methoden ausgeliefert die den Zugriff auf XML erleichtern. Will man z.B. auf alle ProcessNode Elemente zugreifen, dann kann man beim Wurzelelement starten, sich das Kind mit dem Namen "ProcessChain" geben lassen und dann über alle Kindelemente iterieren. Aus der Sicht des Softwareentwicklers beinhaltet dieses Vorgehen die Fehlerquelle, dass "ProcessChain" ein Zeichenkette ist. Man könnte sich bei dieser Zeichenkette vertippen, ohne dass beim Kompilieren eine Fehlermeldung ausgegeben wird. Es wäre besser eine Klasse ProcessNode zu haben mit einer Methode getProcessChain. Genau diese Lücke füllt XMLBeans. XMLBeans erzeugt aus dem XML Schema eine Klassenbibliothek die den Zugriff auf Elemente eines PCML Dokuments per Methodenaufruf ermöglichen.

[1] http://www.w3.org/XML

[2] www.w3.org/**XML**/**Schema**

[3] Liegt im Ordner src/de/bund/bfr/knime/pcml/schema im Plugin de.bund.bfr.knime.pcml.

Im Kapitel 3.4 findet sich das Schema von pcml. Siehe auch Screenshot von einem Tree-View (Abb. 6).

2.2 PORTTYP

Um einen Prozessknoten konfigurieren zu können ist es zwingend notwendig zu wissen welche Matrices und Agenzien an den Inports anliegen. Informationen die beim Konfigurieren der Knoten vorliegen werden durch die Port-Spec von Knoten zu Knoten transportiert. Es wurde ein neuer Porttyp PCML-Port entwickelt dessen Port-Spec folgende Attribute beinhaltet:

- Eine Liste von Matrices mit Mengenangabe
- Ein Liste von Agenzien mit Mengenangabe
- Volumen
- Temperatur
- pH
- Druck
- Wasseraktivität a_w

Der PCML-Port selbst beinhaltet das PCML Dokument, die ID des letzten Prozessknotens und die Nummer des Outport der durch die Port-Spec beschrieben wird.

2.3 IMPORT- EXPORT-KNOTEN

Für die Transformation zwischen XML und PCML wurden zwei neue Knoten entwickelt: "XML to PCML" und "PCML to XML" (Abb. 1). Diese beiden Knoten gewährleisten jederzeit die Anbindung an alle Knoten in KNIME, die XML verarbeiten können. Der Knoten "XML to PCML" ermöglicht ein PCML Dokument in einer Prozesskette zu verwenden, das über Zutaten-Knoten oder Kontaminations-Knoten nicht abgebildet werden kann. Dieser Knoten ist insbesondere hilfreich beim Testen von Prozess-Knoten und Report-Knoten. Der Knoten "PCML to XML" ermöglicht die Weiterverarbeitung von PCML durch Standard-KNIME-Knoten. Aus der Kombination der Knoten "PCML to XML" und "XML XPath" lassen sich beliebige Informationen aus dem PCML Dokument extrahieren und z.B. in einem Birt Report darstellen.

Für den Export der innerhalb der PCML-Struktur vorhandenen Datentabellen in das KNIME Standardtabellenformat wurde ein eigener Knoten entwickelt. Der Knoten "PCML to Data Table" kombiniert alle Datentabellen und fügt Spalten für die absolute Zeit, den Namen des Prozess-Knotens und dessen ID hinzu.

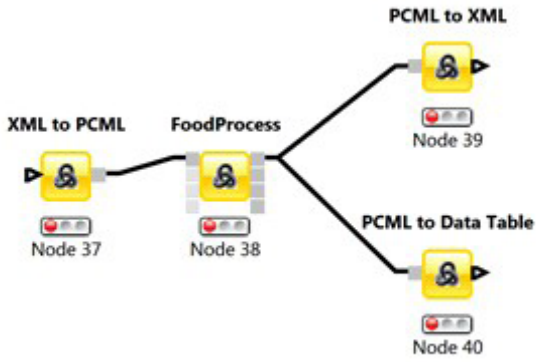

Abb. 1: Neue Knoten: XML to PCML, FoodProcess, PCML to XML, PCMÖ to Data Table

2.4 PROZESS-KNOTEN

Der Prozessknoten umfasst 4 In- und 4 Out-Ports. Nicht alle Ports müssen belegt sein, nur ein In-Port ist obligatorisch …

Der configure-Dialog umfasst

1. alle relevanten Eigenschaften des Gesamtknotens: Prozessname (optional aus dem Katalog der BfR-DB entnehmbar), Kapazität, Dauer, Temperatur, pH, aw und Druck ggf. mit den zugehörigen Eineiten.
2. Für Definitionen bzgl. der In-Ports, Out-Ports und Agenzien sind Tabs vorgesehen:
    1. siehe Abbildung 2:

       In Ports: durch Aktivierung des „Expert-mode“ ist es möglich, Standardparameter für alles „Material“ (=Agenzien und Matrices), das durch den Port fließen zu definieren. Das kann auch Zeitverläufe umfassen, die speziell für die „Materialien“ des jeweiligen In Ports gilt.

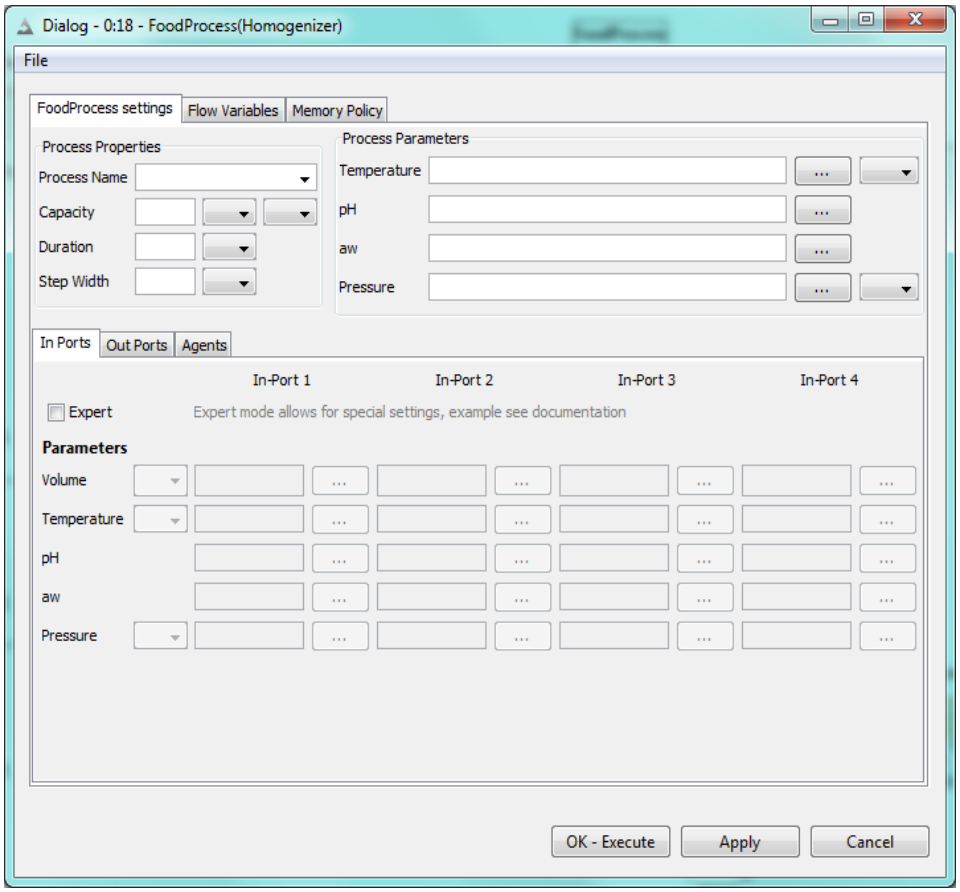

Abb. 2: Configure Dialog des Prozessknotens – In Ports

2. siehe Abbildung 3:

   Hier können Neudefinition von Matrices vorgenommen werden (Beispiel: Erdbeer und Joghurt kann zu Erdbeerjoghurt umdefiniert werden). Weiterhin muss hier definiert werden wie sich das Material bzgl. Volumen auf die jeweiligen Outports aufteilt (OutFlux).

   Out Ports: durch Aktivierung des „Expert-mode“ ist es möglich, Standardparameter für alles „Material“ (=Agenzien und Matrices), das durch den Port fließen zu definieren. Das kann auch Zeitverläufe umfassen, die speziell für die „Materialien“ des jeweiligen Out Ports gilt. Die Zusammensetzung der Materialien des Outport werden definiert durch „Recipe“ (noch zu bauen) und einer eventuellen Neudefinition der Matrix (siehe oben).

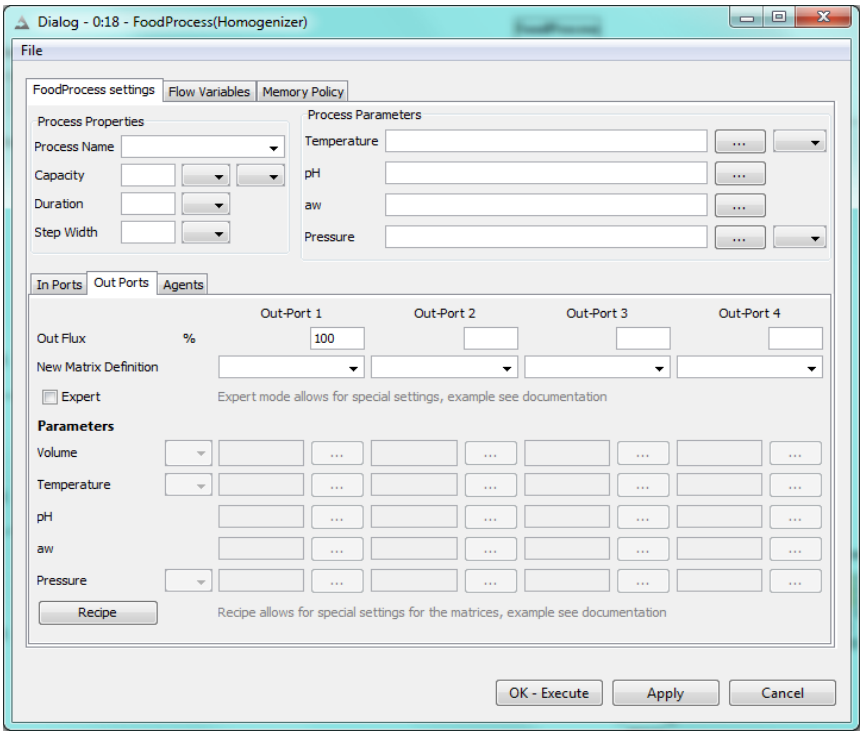

Abb. 3: Configure Dialog des Prozessknotens – Out Ports

3. siehe Abbildung 4:

   es gibt zwei Möglichkeiten die Konzentrationen der Agenzien zu definieren:

   1. „Guess from Recipe“, sollte eigentlich besser „Default“ heissen. Grundannahme ist, dass es zu einer perfekten Mischung aller „Materialien“ gekommen ist und damit die Verteilung gleichverteilt ist.
   2. Individuell kann eingestellt werden in welchen Relationen jedes Agens durch welchen Out-Port fließt.

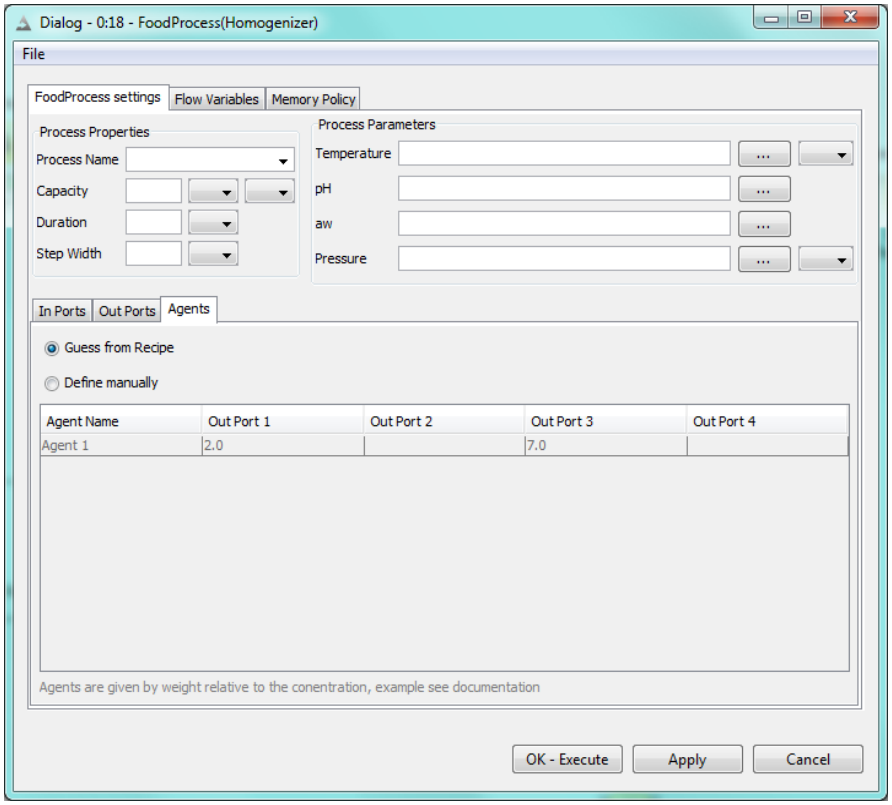

Abb. 4: Configure Dialog des Prozessknotens – Agent Definition

Der Prozessknoten hat zwei Views:

1. Process Chain Port: zeigt die jeweiligen Anteile aller Matrices und Agenzien und die dort gültigen Prozessparameter an, siehe Abbildung 5.
2. PCML model: zeigt PCML in einer Baum-Struktur an, siehe Abbildung 6.

Properties of the Outport

Matrix:	**Vormilch** (110) : 100%
Agent:	No agents defined.
Volume:	1.000.000
Temperature:	n/a
Pressure:	n/a
pH value:	n/a
aw value:	n/a

Abb. 5: OutPort View des Prozessknotens – Die Werte im Out Port sind bereits nach dem configure verfügbar

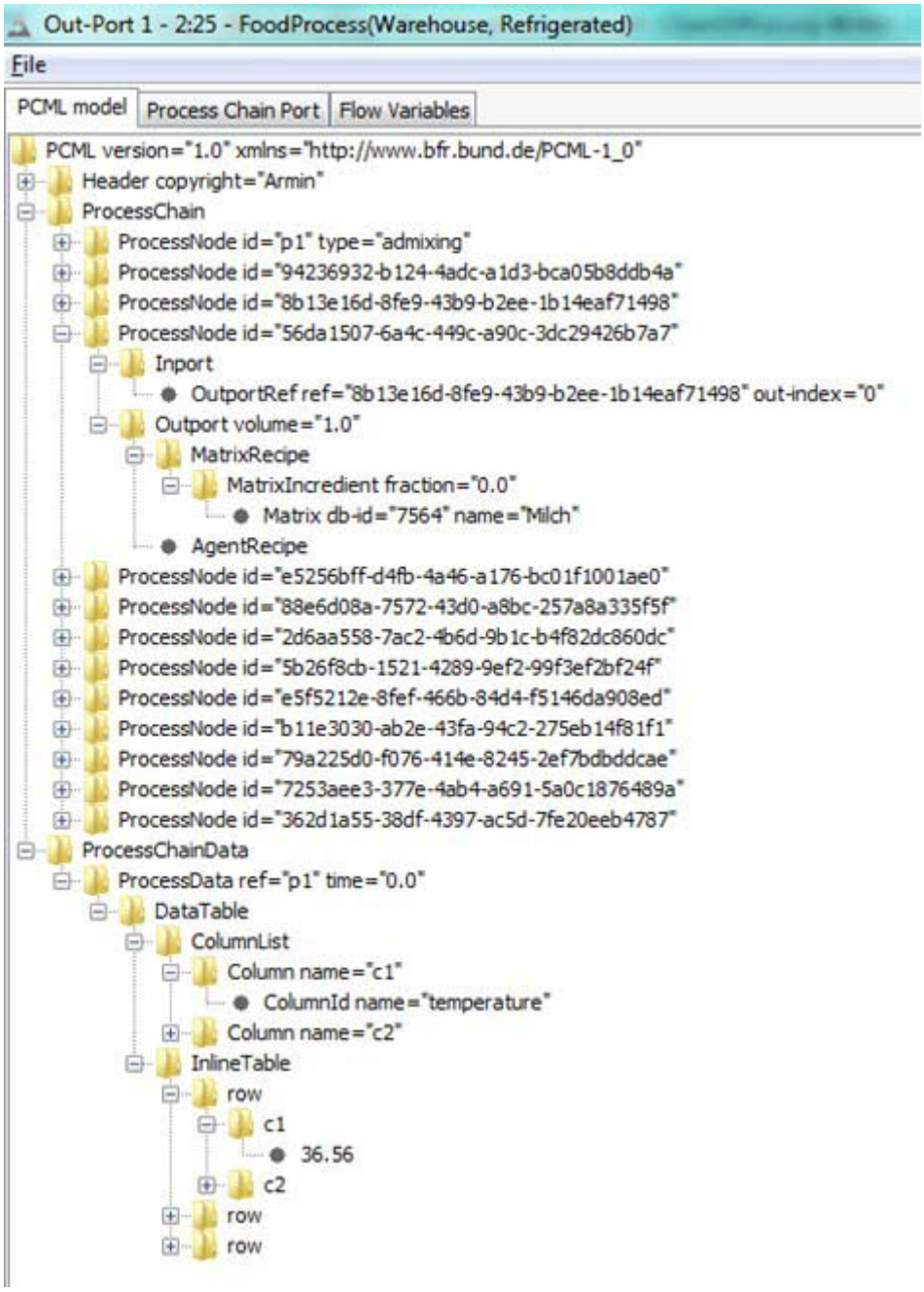

Abb. 6: OutPort View des Prozessknotens – PCML Tree

2.5 AUTOMATISCHE ERSTELLUNG VON WORKFLOWS

Es wurde beispielhaft demonstriert wie eine Workflow aus Java erzeugt werden kann (siehe Quelltext in Kapitel 3.3). Empfohlen wurde diese Funktionalität in den File>Import Dialog von KNIME zu integrieren. Es wird darauf hingewiesen, dass diese Verwendung der Funktionalität von KNIME nicht explizit

unterstützt wird. Es kann durchaus sein, dass in zukünftigen Versionen von KNIME, diese Funktionalität nicht mehr vorhanden ist. Ziel für das BfR ist es hier die Möglichkeit zu haben bereits in einer Datenbank vorhandene Prozessketten automatisch importieren zu können.

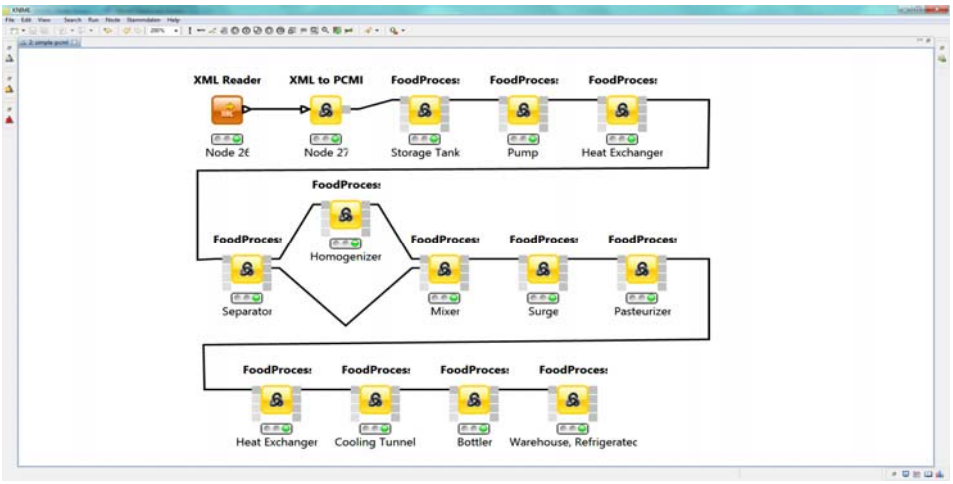

Abb. 7: Beispiel-Workflow einer Prozesskette

2.6 SONSTIGES / PROBLEME / LÖSUNGEN / AUSBLICK

2.6.1 KNOTENBEZEICHNUNG

Prinzipiell gibt es folgende Problematik: Sobald jemand einen neuen Workflow=Prozesskette manuell generiert, wird der Knotenname erstmal nur „node n“ genannt. Der definierte Prozessname im configure-Dialog wird nicht vollautomatisch auch für den Knotennamen übernommen, das ist in KNIME so nicht möglich. Es ist aber angedacht einen „Refresh-Button“ für diesen Zweck im Menü anzubieten, parallel zu der Möglichkeit Workflows automatisch zu erzeugen. Es ist für den Nutzer natürlich sehr hilfreich zu wissen, welcher Prozessschritt ein Knoten darstellt ohne extra den Configure-Dialog öffnen zu müssen. Das ist im übrigen eine Funktionalität, die allen KNIME Knoten gut zu Gesicht stehen würde – eine Art „Default“-Bezeichnung des Knotens, die der Workflowersteller beliebig definieren kann.

2.6.2 LOCALE

Weiteres Problem: Locale: die Standard-Darstellung der Zahlen (US) in den Configure Dialogen ist verwirrend, der Nutzer soll die Möglichkeit haben seiner Gewohnheit gerecht zu werden.

Eine Lösung ist folgendermaßen angedacht: Es wird eine Locale Einstellmöglichkeit im FoodProcess Knoten geben, der dann automatisch von allen FoodProcess Knoten übernommen wird. Technisch wird das realisiert durch eine Setting in Eclipse.

In PCML abgespeichert wird allerdings immer nur mittels Locale.US.

2.6.3 PERFORMANCE

Sollte es vorkommen, dass es zu Performance Problemen kommt, weil der Inline Table in der pcml-Struktur sehr groß wird, z.B. über 100000 Zeilen beinhaltet, dann gibt es die Möglichkeit die Table als Internals in

KNIME abzuspeichern und via TableLocator in der pcml-Struktur darauf zu verweisen.

2.6.4 FORMELN ↔ ZAHLEN

Da alle Parameter über Formeln abgespeichert werden können, werden die zugehörigen Werte nicht als Double abgespeichert, sondern als String, auch wenn es sich um Konstanten handelt. Der String kann von einem Parser gelesen werden:
http://math.hws.edu/javamath/javadoc/edu/hws/jcm/data/Parser.html.

2.6.5 WEITERE NOCH ZU BAUENDE KNOTEN

KONTAMINATIONSKNOTEN

- Configure
  - Ein oder mehrere Agenzien, Freitext, optional: aus einem Katalog der BfR-DB auswählbar
  - Masse/Volumen/Konzentration angeben
  - Verschiedene Einheiten auswählbar
  - evtl. Zeitpunkt der Zugabe

ZUTATENKNOTEN

- Configure
  - Ein oder mehrere Matrices, Freitext, optional: aus einem Katalog der BfR-DB auswählbar
  - Masse/Volumen angeben
  - Verschiedene Einheiten auswählbar
  - Temperatur, pH, aw der Zutat definierbar
  - evtl. Zeitpunkt der Zugabe

BERECHNUNGSKNOTEN

- Input
  - PCML-Struktur
  - Standard-KNIME Tabelle
  - optional Modell (Formel, pmml?)
- Configure
  - optional: Prozessschritt ID, für den berechnet werden soll
  - optional: Eigene Formel definierbar
  - optional: Auswahl eines Modells aus der BfR_DB
    (Suchkriterien Temp, pH, aw, Druck, Matrix, Agens)
    vielleicht besser als Extraknoten, der seinen Output
    dann über den pmml Port reinleitet?!?

- Execute
  - Neu-Berechnung der Konzentrationen für
    - den letzten Prozessschritt oder
    - die definierte Prozessschritt-ID basierend auf
    - pmml oder
    - Formel oder
    - DB
- Output
  - PCML-Struktur
  - Standard-KNIME Tabelle

REPORTKNOTEN

- Nutzung des BIRT Report Frameworks

3 TECHNISCHES

3.1 XMLBEANS

Unten aufgelistete Befehlen erzeugen eine Klassenbibliothek die den Zugriff auf Elemente eines PCML Dokuments per Methodenaufruf ermöglicht. Es wird XMLBeans in der Version 2.4.0 verwendet.

```
set PATH=%PATH%;C:\Program Files (x86)\Java\jdk1.6.0_18\bin
set JAVA_HOME=C:\Program Files (x86)\Java\jre6

set XMLBEANS_HOME=<Pfad zum Ordner xmlbeans-2.4.0>
set XMLBEANS_LIB=%XMLBEANS_HOME%\lib
set PATH=%PATH%;%XMLBEANS_HOME%\bin
set
CLASSPATH=%CLASSPATH%;%XMLBEANS_HOME%/lib/xbean.jar;%XMLBEANS_HOME%/lib
/jsr173_1.0_api.jar

set MY_ECLIPSE_WORKSPACE=<Pfad zum Eclipse Workspace>
cd %MY_ECLIPSE_WORKSPACE%\de.bund.bfr.knime.pcml\lib

scomp -src src -out xmlbeans-pcml-1-0.jar
%MY_ECLIPSE_WORKSPACE%\de.bund.bfr.knime.pcml\src\de\bund\bfr\knime\pcm
l\schema\pcml-1-0.xsd
```

3.2 DATENBANK ALS ECLIPSE PLUGIN EINBINDEN

Um die BfR-interne Datenbank optional nutzbar zu machen, kann sie als Plugin ausgeliefert werden und innerhalb der Knoten angesprochen werden. Die Datenbank selbst und alle benötigten Klassenbibliotheken werde in den Ordner lib des Plugins kopiert. Der Zugriff erfolgt über die vorhandenen Klassen einzig die Konfiguration in den KNIME Preferences ist nicht mehr obligatorisch. Ist in den KNIME Preferences in der Kategorie BfR ein Pfad zu einer Datenbank gesetzt und ein Benutzer mit Password definiert, so werden diese Einstellungen verwendet, ohne Konfiguration wird auf die mitgelieferte Datenbank mit dem Standard-Benutzer zugegriffen. Der Pfad zur Datenbank ergibt sich aus dem absoluten Pfad zum Plugin und dem relativen Pfad lib/xx.jar. Z.B. wird der Pfad zu lib/BfR_DB_1.3.6.jar mit folgender Anweisung ermittelt:

```
1 // Get the bundle this class belongs to.
2 Bundle bundle = FrameworkUtil.getBundle(this.getClass());
3 try {
4     // Get the URL to the file
5     URL incURL = FileLocator.find(bundle,
6             new Path("/lib/BfR_DB_1.3.6.jar"), null);
7     // Create a file object from the URL
8     File incFile = new File(FileLocator.toFileURL(incURL).getPath());
9     // TODO: Process file
10 } catch (IOException e) {
11     throw new IllegalStateException(
12             "Cannot locate necessary libraries.", e);
13 }
```

3.3 LOMBOK

Das Projekt Lombok[4] verfolgt das Ziel typischen Java Quelltext lesbarer zu gestalten und darüber hinaus Programmierfehler zu vermeiden. Lombok definiert z.B. die Annotationen @Getter und @Setter durch die Methoden zum Lesen und Schreiben von Klassenvariablen erzeugt werden. Sehr nützlich ist auch die @EqualsAndHashCode Annotation die eine korrekte Implementierung der Methoden equals und hashCode garantiert. Alle Annotationen sind auf http://projectlombok.org/features/index.html aufgelistet.
Um Lombok in Eclipse verwenden zu können, muss man lombok.jar installieren (java -jar lombok.jar oder Doppelklick) und es zum Classpath der Eclipse Plugins hinzufügen.

[4] http://projectlombok.org/features/index.html

3.4 AUTOMATISCHES ERZEUGEN VON KNIME WORKFLOWS MIT KNIME 2.5.4.

Die Methode createWorkflow erzeugt ein Projekt und fügt dem Workflow einige Knoten hinzu. Die Knoten werden mit dem Autolayout platziert.

```
 1 private void crateWorkflow(final String projectName) throws Exception {
 2     // create wfm for the project
 3     final WorkflowManager wfm = WorkflowManager.ROOT
 4                     .createAndAddProject(projectName);
 5     // add nodes and connect them
 6     addNodes(wfm);
 7
 8     if (wfm.getNodeContainers().size() > 1) {
 9         // Autolayout workflow
10         AutoLayoutCommand layoutCommand = new AutoLayoutCommand(wfm,
null);
11         layoutCommand.execute();
12     }
13
14     // create project
15     IProject proj = createProject(new Path(projectName));
16     // save workflow to the project
17     wfm.save(proj.getLocation().toFile(), new ExecutionMonitor(),
false);
18     // refresh so that project shows up in "Workflow projects"
19     proj.refreshLocal(IResource.DEPTH_INFINITE, new
NullProgressMonitor());
20
21     // remove lock from the project
22     VMFileLocker.unlockForVM(proj.getLocation().toFile());
23
24     // get the "workflow file"
25     final IFile workflowFile = proj.getFile(
26                     new Path(WorkflowPersistor.WORKFLOW_FILE));
27     Display display = Display.getDefault();
28     display.asyncExec(new Runnable() {
29         @Override
30                     public void run() {
31             IWorkbenchPage page =
32
PlatformUI.getWorkbench().getActiveWorkbenchWindow()
```

```
33                              .getActivePage();
34              try {
35                      // open the workflow
36                  IDE.openEditor(page, workflowFile, true);
37              } catch (PartInitException e) {
38                  // ignore it
39              }
40
41          }
42      });
43  }
44
45  /** Add nodes the the workflow manager and connect them. */
46  private void addNodes(final WorkflowManager wfm) {
47      FoodProcessNodeFactory pf = new FoodProcessNodeFactory();
48      // create a node with default settings
49      NodeID p1 = wfm.createAndAddNode(pf);
50      // set dummy location, the location will later be change by the
51      // layout manger, but a location must be set here, so that the node
52      // is correctly initialized.
53      setDummyLocation(wfm, p1);
54
55      // create another node
56      NodeID p2 = wfm.createAndAddNode(pf);
57      setDummyLocation(wfm, p2);
58
59      // get the container of the node
60      SingleNodeContainer p2Container =
(SingleNodeContainer)wfm.getNodeContainer(p2);
61      // set the node description, normally set by clicking "set
description"
62      // in the context menu of the node.
63      p2Container.setCustomDescription("Ein schöner Milchturm");
64
65      // create a custom node annotation (diplayed at the bottom of the
node)
66      AnnotationData p2AnnoData =
NodeAnnotationData.createFromObsoleteCustomName("Milchturm");
67      // set custom node annotation
68      p2Container.getNodeAnnotation().getData().copyFrom(p2AnnoData,
true);
69      // set dirty so that changer are saved
```

```
 70      p2Container.setDirty();
 71
 72
 73      // get the node from the container
 74      Node p2Node = p2Container.getNode();
 75
 76      // create custom settings for the food process node
 77      FoodProcessSetting foodProcessSetting = new FoodProcessSetting(4,
4);
 78      foodProcessSetting.setProcessName("Milchturm");
 79
 80      FoodProcessNodeSettings p2Settings = new FoodProcessNodeSettings();
 81      p2Settings.setFoodProcessSetting(foodProcessSetting);
 82
 83      // get the node model
 84      FoodProcessNodeModel p2Model =
((FoodProcessNodeModel)p2Node.getNodeModel());
 85      // set the custom settings
 86      p2Model.setSetting(p2Settings);
 87
 88
 89      // Create another node
 90      NodeID p3 = wfm.createAndAddNode(pf);
 91      setDummyLocation(wfm, p3);
 92
 93      // Connect the nodes
 94      // Port 0 is the "Flow Variable Port"
 95      // Port 1 is the first normal port
 96      wfm.addConnection(p1, 1, p2, 1);
 97      wfm.addConnection(p1, 2, p3, 1);
 98
 99
100      // Create a SimpleDatabaseReaderNode
101      SimpleDatabaseReaderNodeFactory vcnf2 = new
SimpleDatabaseReaderNodeFactory();
102      NodeID databaseReaderNID = wfm.createAndAddNode(vcnf2);
103      setDummyLocation(wfm, databaseReaderNID);
104  }
105
106
107  /** Place the node a (100px, 100px); */
108  private void setDummyLocation(final WorkflowManager wfm, final NodeID
```

```
nodeId) {
109         wfm.getNodeContainer(nodeId).setUIInformation(
110                         new NodeUIInformation(100, 100, -1, -1,
false));
111 }
112
113 /** Create a Project. */
114 private IProject createProject(final IPath workflowPath) throws
CoreException {
115         IProgressMonitor monitor = new NullProgressMonitor();
116
117          IWorkspaceRoot root =
ResourcesPlugin.getWorkspace().getRoot();
118         IResource resource = root.findMember(workflowPath);
119         if (resource != null) {
120             throwCoreException("Resource \"" + workflowPath.toString()
121                     + "\" does already exist.", null);
122         }
123         // check if there is a folder with the same name on the file
system
124         // see bug (http://bimbug.inf.uni-
konstanz.de/show_bug.cgi?id=1912)
125         IPath rootLocation = root.getLocation();
126         if (rootLocation != null) {
127             IPath absolutePath = rootLocation.append(workflowPath);
128             if (absolutePath.toFile().exists()) {
129                 throwCoreException(
130                         "Resource " + workflowPath + " already
exists!", null);
131             }
132         }
133         ContainerGenerator generator = new
ContainerGenerator(workflowPath);
134         IContainer containerResult =
generator.generateContainer(monitor);
135         if (containerResult instanceof IProject) {
136             IProject project = (IProject)containerResult;
137             // open the project
138             project.open(monitor);
139             // Create project description, set the nature IDs and
build-commands
140             try {
```

```
141                 // set the nature id of the project is enough
142                 // the name is already set by IProject#create()
143                 IProjectDescription description =
project.getDescription();
144                 description.setNatureIds(new
String[]{KNIMEProjectNature.ID});
145                 project.setDescription(description, monitor);
146
147             } catch (CoreException ce) {
148                 throwCoreException(
149                         "Error while creating project description for "
150                         + project.getName(), ce);
151             }
152             return project;
153         }
154         return null;
155 }
156
157 private static void throwCoreException(final String message,
158         final Throwable t)
159     throws CoreException {
160     IStatus status =
161             new Status(IStatus.ERROR, "de.bund.bfr.ui", IStatus.OK,
162                     message, t);
163     throw new CoreException(status);
164 }
```

3.5 XML SCHEMA – PCML-1-0.XSD

```
 1 <xs:schema
 2   xmlns:xs="http://www.w3.org/2001/XMLSchema"
 3   targetNamespace="http://www.bfr.bund.de/PCML-1_0"
 4   xmlns="http://www.bfr.bund.de/PCML-1_0"
 5   elementFormDefault="qualified">
 6   <xs:element name="PCML">
 7     <xs:complexType>
 8       <xs:sequence>
 9         <xs:element ref="Header" />
10         <xs:element ref="ProcessChain"  minOccurs="0" maxOccurs="1" />
11         <xs:element ref="ProcessChainData"  minOccurs="0" maxOccurs="1"
/>
12         <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"
/>
```

```
13        </xs:sequence>
14        <xs:attribute name="version" type="xs:string" use="required" />
15      </xs:complexType>
16    </xs:element>
17    <xs:element name="Header">
18      <xs:complexType>
19        <xs:sequence>
20          <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"
/>
21          <xs:element minOccurs="0" ref="Application" />
22          <xs:element minOccurs="0" maxOccurs="unbounded"
ref="Annotation" />
23          <xs:element minOccurs="0" ref="Timestamp" />
24        </xs:sequence>
25        <xs:attribute name="copyright" type="xs:string" use="required" />
26        <xs:attribute name="description" type="xs:string" />
27      </xs:complexType>
28    </xs:element>
29    <xs:element name="Application">
30      <xs:complexType>
31        <xs:sequence>
32          <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"
/>
33        </xs:sequence>
34        <xs:attribute name="name" type="xs:string" use="required" />
35        <xs:attribute name="version" type="xs:string" />
36      </xs:complexType>
37    </xs:element>
38    <xs:element name="Annotation">
39      <xs:complexType mixed="true">
40        <xs:sequence>
41          <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"
/>
42        </xs:sequence>
43      </xs:complexType>
44    </xs:element>
45    <xs:element name="Timestamp">
46      <xs:complexType mixed="true">
47        <xs:sequence>
48          <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"
/>
49        </xs:sequence>
```

```
50      </xs:complexType>
51    </xs:element>
52
53    <xs:complexType name="NameAndDatabaseId">
54      <xs:attribute name="name" type="xs:string" use="required" />
55      <xs:attribute name="db-id" type="xs:int" />
56    </xs:complexType>
57
58    <xs:element name="ProcessChain">
59      <xs:complexType>
60        <xs:sequence minOccurs="0" maxOccurs="unbounded">
61          <xs:element ref="ProcessNode" />
62          <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"
/>
63        </xs:sequence>
64      </xs:complexType>
65    </xs:element>
66    <xs:element name="ProcessNode">
67      <xs:complexType>
68        <xs:sequence>
69          <xs:element name="Process" type="NameAndDatabaseId" />
70          <xs:element name="Parameters" type="ProcessParameters" />
71          <xs:element ref="Inport" minOccurs="0" maxOccurs="4" />
72          <xs:element ref="Outport" minOccurs="1" maxOccurs="4" />
73        </xs:sequence>
74        <xs:attribute name="id" type="xs:string" use="required" />
75        <xs:attribute name="type" type="ProcessNodeType" />
76      </xs:complexType>
77    </xs:element>
78
79    <xs:simpleType name="ProcessNodeType">
80      <xs:restriction base="xs:string">
81        <xs:enumeration value="processing" />
82        <xs:enumeration value="admixing" />
83        <xs:enumeration value="contaminating" />
84      </xs:restriction>
85    </xs:simpleType>
86
87    <xs:complexType name="ProcessParameters">
88      <xs:sequence>
89        <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"
/>
```

```
    </xs:sequence>
    <xs:attribute name="capacity" type="xs:double" />
    <xs:attribute name="duration" type="xs:double" use="required" />
    <xs:attribute name="step-width" type="xs:double" />
    <xs:attribute name="temperature" type="xs:string" />
    <xs:attribute name="pH" type="xs:string" />
    <xs:attribute name="aw" type="xs:string" />
    <xs:attribute name="pressure" type="xs:string" />
  </xs:complexType>
  <xs:element name="Inport">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="OutportRef" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="OutportRef">
    <xs:complexType>
      <xs:attribute name="ref" type="xs:string" use="required" />
      <xs:attribute name="out-index" type="xs:int" use="required" />
    </xs:complexType>
  </xs:element>
  <xs:element name="Outport">
    <xs:complexType>
      <xs:sequence>
        <!-- for new matrix definition -->
        <xs:element name="Matrix" type="NameAndDatabaseId"
          minOccurs="0" maxOccurs="1" />
        <xs:element ref="MatrixRecipe" minOccurs="0" maxOccurs="1" />
        <xs:element ref="AgentRecipe" minOccurs="0" maxOccurs="1" />
        <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"
/>
      </xs:sequence>
      <xs:attribute name="volume" type="xs:string" use="required" />
      <xs:attribute name="temperature" type="xs:string" />
      <xs:attribute name="pH" type="xs:string" />
      <xs:attribute name="aw" type="xs:string" />
      <xs:attribute name="pressure" type="xs:string" />
    </xs:complexType>
  </xs:element>
  <xs:element name="MatrixRecipe">
    <xs:complexType>
```

```
131       <xs:sequence>
132         <xs:element ref="MatrixIncredient" minOccurs="1"
133           maxOccurs="unbounded" />
134         <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"
/>
135       </xs:sequence>
136     </xs:complexType>
137   </xs:element>
138   <xs:element name="MatrixIncredient">
139     <xs:complexType>
140       <xs:sequence>
141         <xs:element name="Matrix" type="NameAndDatabaseId" />
142       </xs:sequence>
143       <xs:attribute name="fraction" type="xs:double" />
144     </xs:complexType>
145   </xs:element>
146   <xs:element name="AgentRecipe">
147     <xs:complexType>
148       <xs:sequence>
149         <xs:element ref="AgentIncredient" minOccurs="1"
150           maxOccurs="unbounded" />
151         <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"
/>
152       </xs:sequence>
153     </xs:complexType>
154   </xs:element>
155   <xs:element name="AgentIncredient">
156     <xs:complexType>
157       <xs:sequence>
158         <xs:element name="Agent" type="NameAndDatabaseId" />
159       </xs:sequence>
160       <xs:attribute name="quantity" type="xs:double" />
161     </xs:complexType>
162   </xs:element>
163
164   <xs:element name="Extension">
165     <xs:complexType>
166       <xs:complexContent mixed="true">
167         <xs:restriction base="xs:anyType">
168           <xs:sequence>
169             <xs:any processContents="skip" minOccurs="0"
maxOccurs="unbounded" />
```

```
170           </xs:sequence>
171           <xs:attribute name="extender" type="xs:string" use="optional"
/>
172           <xs:attribute name="name" type="xs:string" use="optional" />
173           <xs:attribute name="value" type="xs:string" use="optional" />
174         </xs:restriction>
175       </xs:complexContent>
176     </xs:complexType>
177   </xs:element>
178
179   <xs:element name="ProcessChainData">
180     <xs:complexType>
181       <xs:sequence minOccurs="0" maxOccurs="unbounded">
182         <xs:element ref="ProcessData" />
183         <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"
/>
184       </xs:sequence>
185     </xs:complexType>
186   </xs:element>
187   <xs:element name="ProcessData">
188     <xs:complexType>
189       <xs:sequence>
190         <xs:element ref="DataTable" />
191       </xs:sequence>
192       <!-- Reference to the ProcessNode element. -->
193       <xs:attribute name="ref" type="xs:string" use="required" />
194       <xs:attribute name="time" type="xs:double" />
195     </xs:complexType>
196   </xs:element>
197   <xs:element name="DataTable">
198     <xs:complexType>
199       <xs:sequence>
200         <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"
/>
201         <xs:element ref="ColumnList" />
202         <xs:choice>
203           <xs:element ref="TableLocator" />
204           <xs:element ref="InlineTable" />
205         </xs:choice>
206       </xs:sequence>
207       <xs:attribute name="name" type="xs:string" />
208     </xs:complexType>
```

```
209   </xs:element>
210   <xs:element name="ColumnList">
211     <xs:complexType>
212       <xs:sequence>
213         <xs:element ref="Column" minOccurs="1" maxOccurs="unbounded" />
214       </xs:sequence>
215     </xs:complexType>
216   </xs:element>
217   <xs:element name="Column">
218     <xs:complexType>
219       <xs:sequence>
220         <xs:choice>
221           <xs:element name="Matrix" type="NameAndDatabaseId" />
222           <xs:element name="Agent" type="NameAndDatabaseId" />
223           <xs:element name="ColumnId" type="NameAndDatabaseId" />
224         </xs:choice>
225       </xs:sequence>
226       <xs:attribute name="name" type="xs:string" />
227     </xs:complexType>
228   </xs:element>
229   <xs:element name="TableLocator">
230     <xs:complexType>
231       <xs:sequence>
232         <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"
/>
233       </xs:sequence>
234     </xs:complexType>
235   </xs:element>
236   <xs:element name="InlineTable">
237     <xs:complexType>
238       <xs:sequence>
239         <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"
/>
240         <xs:element ref="row" minOccurs="0" maxOccurs="unbounded" />
241       </xs:sequence>
242     </xs:complexType>
243   </xs:element>
244   <xs:element name="row">
245     <xs:complexType>
246       <xs:complexContent mixed="true">
247         <xs:restriction base="xs:anyType">
248           <xs:sequence>
```

```
249             <xs:any processContents="skip" minOccurs="1"
maxOccurs="unbounded" />
250           </xs:sequence>
251         </xs:restriction>
252       </xs:complexContent>
253     </xs:complexType>
254   </xs:element>
255 </xs:schema>
```

4 ÜBERGABE

Übergeben wurden:

- Sourcen der Plugins de.bund.bfr.knime.pcml und de.bund.bfr.knime.util
- Beispiel xml-Dateien
- Source Code für Workflow Generator
- xml-beans-2.4.0.jar
- Dokumentation
- Alle Rechte an der, im Berichtszeitraum während der Erfüllung des Auftrags erzeugten Software verbleiben ausschliesslich und vollständig beim Auftraggeber (BfR). Die Rechte an Änderungen bestehender Software sind beim ursprünglichen Autor und beim Auftraggeber. Durch die Ausführung des Auftrags erwirkt die KNIME.com AG oder ihre Beauftragten keine Rechte am entstandenen Werk.

KNIME.com AG
Technoparkstrasse 1
8005 Zürich, Switzerland
http://www.KNIME.com
contact@KNIME.com