UC0005

-  
VSOS - Virtuelle Server Order System

**Version:**

**Last changed:** 05.03.2020

NTT Data Services Germany GmbH

Revision History

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Version | Effective Date (DD/MM/YYYY) | Brief Description of Change | Affected Section(s) | Prepared By | Reviewed By | Approved By |
| 1.0 | 28/02/2020 | Initial draft | all | Roland Höhl | Roland Höhl | n/a |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

# Description:

Viele der NTT Data Services Kunden (BMW, T-Systems, Deutsche Telekom etc.) nutzen heute interne oder externe Server Farmen um ihre Entwicklung, Test- und/oder Produktionsumgebungen flexibel aufzubauen und zu verwalten (Cloud Services).

Leider ist die Bereitstellung von virtuellen Servern aus dem Pool vorhandener dedizierter Server immer noch ein langwieriger Prozess. Endkunden verweisen immer wieder darauf, dass die Bereitstellung einer virtuellen Server Instanz bei Amazon nur 1 Tage dauert, jedoch in den internen Rechenzentren der Bereitstellungsprozess bis zu 6 Monaten dauern kann. Hintergründe sind die langen Bereitstellungsprozesse (Rückfragen beim Endkunden was er eigentlich genau braucht), Kapazitätsplanungen (wir bauen erst wenn wir einen Server voll haben …) und noch vieles mehr.

Die Zielsetzung dieses internen Entwicklungsprojektes ist dem Endanwender ein leicht zu verstehendes Interface zur Verfügung zu stellen: Die Funktionen sollen sein:

* Virtuelle Instanzen selbst konfigurieren (Einfaches Userinterface)
  + OS Typ, CPU, RAM, Storage, preinstalled Software, User Accounts etc.
* Instanzen zum Provisionieren freigeben und zum Aufbau freigeben.
  + Kosten Vorkalkulation
* Instanzen löschen und De-Kommissionierung.
* Vorhandene freie Serverkapazitäten anzeigen, um daraus noch neue Konfigurationen zu schneiden („Vielleicht ist es nicht mehr so viel wie du gerne hättest, aber vielleicht langt das was noch da ist“)
* Anzeige der virtuellen Instanzen die dem Anwender zugeordnet sind inkl. Storage Nutzung, CPU und Netzauslastung für die Provisionierung.
* Falls keine freien Ressourcen auf den Servern vorhanden sind sollen Informationen gesammelt werden um neue Server (physikalische) zu bestellen.

Im Hintergrund soll ein Auftrag erstellt werden, der automatisiert:

* Die Parameter für die virtuelle Instanz zusammenstellt und dann an einen im Backend-Prozess (AWS, Azure, T-Systems, BMW Cloud) zum Bauen zusendet
* Der Backend-Prozess soll dann die entsprechenden Konfigurationen auswerten und über die entsprechende Cloud API das Bauen der Instanz anstoßen.
* Success/Error/Warnings etc. aus der API sind dem Endanwender / Administrator zuzustellen (Mail etc.)
* Konfigurationen sollen gespeichert werden um sie wieder zu verwenden
* Auf Anfrage sind die Auslastungen und Provisionierungsdaten dem Endanwender zur Verfügung zu stellen (Abrechnung etc.)

Die Zielsetzung für den Cloud Anbieter ist es frühzeitig korrekte, standardisierte Konfigurationen zu erhalten auf die Server Kapazitäten bereitgestellt werden.

## System Landscape

Als Benutzerinterface soll eine Web Oberfläche gebaut werden. Als Basis soll ein Java Tomcat Server dienen der die Frontend und Backend Prozesse bereitstellen kann.

Die Software (Frontend / Backend) soll in Java implementiert werden. Als Entwicklungsplattform soll Netbeans zum Einsatz kommen.

Als Datenbank, zum Speichern der Konfigurationen, Benutzerverwaltungen etc. soll eine PostgreSQL Datenbank zum Einsatz kommen. Der Backend Prozess soll flexibel aufgebaut sein so dass der Cloud Anbieter (AWS, Azure, T-Systems…) ausgetauscht werden kann.

Das Projekt und die Sourcen sollen in github ([www.github.com](http://www.github.com)) verwaltet werden, um einen gemeinsamen Repository Zugriff zu gewährleisten.

In der Entwicklungsphase soll die Amazon Cloud (AWS) als erste Plattform angebunden werden. Zu einem späteren Zeitpunkt kann dann Azure integriert werden.

Die Entwicklung findet lokal auf den Rechnern statt.

Github dient als Source Repository zum Verwalten des Sourcecodes.

Als Entwicklungsumgebung kann Linux oder Windows zum Einsatz kommen.

Als Test Server steht eine AWS Instanz zur Verfügung. Auf dieser sind bisher installiert:

* Ubuntu 18.04 LTS Linux, Apache TomCat 6 und Oracle 12g Datenbank
* PostgreSQL Datenbank OpenProject Server (10.1)

<properties path=’${project\_loc}/template/conf.properties’ /> <drop/>

<config>

<output path=’${outputPath}’ />

</config>

<context model=’${model}’ element=’{0}’ importedBundles=’gmf;papyrus’/>

<context model=’ D:\Development\UC0005\UC0005.notation' element=’{0}’ importedBundles='gmf;papyrus' />

<gendoc><drop/>

# Wireframe Ablauf mit Pencil Project

<Beschreibung> TBD

## Diagramme und Screens

<Beschreibung> TBD

# Fachliche Datenmodelle

<Beschreibung> TBD

## Beschreibung

<Beschreibung> TBD

# Technische Diagramme:

[for (d : notation::Diagram |notation::Diagram.allInstances()->sortedBy(name))]<drop/>

## [d.name/]

<image object='[d.getDiagram()/]' maxW='true' keepH='false' keepW=’false’ ><drop/>

</image>

*Figure: [d.name.clean()/]*

[/for]<drop/>

</gendoc><drop/>

<context model=’ D:\Development\UC0005\UC0005.uml’ element=’{0}’ importedBundles='gmf;papyrus' />

<gendoc><drop/>

# Objekt description

[for (cl:Class | self.eAllContents(Class)->sortedBy(name))]<drop/>

## [cl.name/]

**Documentation:**

[for (co:Comment | cl.ownedComment)]<drop/>

<dropEmpty>[co.\_body.cleanAndFormat()/]</dropEmpty>

[/for]<drop/>

[if (cl.oclAsType(uml::Classifier).general->notEmpty())]<drop/>

**Parent class:** [cl.oclAsType(uml::Classifier).general ->asSequence()->first().name/]

[/if]<drop/>

**Applied stereotypes:**

[for (st:Stereotype | cl.getAppliedStereotypes())]<drop/>

* [st.name/]

[for (oa:Property|st.ownedAttribute)]<drop/>

* [if (not oa.name.contains('base'))][oa.name/]: [if (not cl.getValue(st, oa.name).oclIsUndefined())][if oa.name.contains('condition')][cl.getValue(st, oa.name).oclAsType(String)/] [else][cl.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/][/if][else]<drop/>[/if]

[/if] <drop/>

[/for]<drop/>

[/for]<drop/>

[if cl.ownedAttribute->notEmpty()]<drop/>

<table><drop/>

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attribute Name** | **Type** | **Mult.** | **Stereotypes** | **Description** |

[for (p:Property|cl.allAttributes())]<drop/>

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| [p.name/] | [p.type.name/] | [if(p.lower=p.upper)]1[else][p.lower/]..[if(p.upper=-1)]\*[else][p.upper/][/if][/if] | [for (st:Stereotype | p.getAppliedStereotypes())]<drop/>  [st.name/]  [for(oa:Property|st.ownedAttribute)]<drop/>   * [if oa.name.contains('attribute')]AVC: [p.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/]   [else]<drop/>   * [if oa.name.contains('isInvariant')]isInvariant: [p.getValue(st, oa.name).oclAsType(Boolean)/]   [else]<drop/>   * [if oa.name.contains('value')]valueRange: [if (not p.getValue(st, oa.name).oclIsUndefined())][p.getValue(st, oa.name).oclAsType(String).clean()/][else] no range constraint [/if]   [else]<drop/>   * [if oa.name.contains('support')]support: [p.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/]   [else]<drop/>   * [if oa.name.contains('condition')][if (not p.getValue(st, oa.name).oclIsUndefined())]condition:[p.getValue(st, oa.name).oclAsType(String).clean()/][else] <drop/> [/if]   [else]<drop/>   * [if oa.name.contains('passedByRef')] [if (not p.getValue(st, oa.name).oclIsUndefined())][p.getValue(st, oa.name).oclAsType(Boolean)/][else] undefined [/if]   [else]<drop/>   * [if oa.name.contains('reference')][if (not p.getValue(st, oa.name).oclIsUndefined())]reference:[p.getValue(st, oa.name).oclAsType(String).clean()/][else] <drop/> [/if]   [else]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/for]<drop/>  [/for]<drop/> | [for (c:Comment | p.ownedComment)] <drop/>  [c.\_body.clean()/]  [/for] |

[/for]<drop/>

</table><drop/>

[else][/if]<drop/>

[/for]<drop/>

## Datatyps

[for (dt:DataType | self.eAllContents(DataType)->sortedBy(name))]<drop/>

[if dt.oclIsTypeOf(DataType)]<drop/>

### Datatyp: [dt.name/]

[for (co:Comment | dt.ownedComment)]<drop/>

<dropEmpty>[co.\_body.clean()/]</dropEmpty>

[/for]<drop/>

[if dt.ownedAttribute->notEmpty()]<drop/>

<table><drop/>

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Attribute Name** | **Type** | **Mult.** | **Access** | **Stereotypes** | **Description** |

[for (p:Property|dt.allAttributes())]<drop/>

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| [p.name/] | [p.type.name/] | [if(p.lower=p.upper)]1[else][p.lower/]..[if(p.upper=-1)]\*[else][p.upper/][/if][/if] | [if(not(p.isReadOnly))]RW[else]R[/if] | [for (st:Stereotype | p.getAppliedStereotypes())]<drop/>  [st.name/]  [for(oa:Property|st.ownedAttribute)]<drop/>   * [if oa.name.contains('attribute')]AVC: [p.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/]   [else]<drop/>   * [if oa.name.contains('isInvariant')]isInvariant: [p.getValue(st, oa.name).oclAsType(Boolean)/]   [else]<drop/>   * [if oa.name.contains('value')]valueRange: [if (not p.getValue(st, oa.name).oclIsUndefined())][p.getValue(st, oa.name).oclAsType(String).clean()/][else] no range constraint [/if]   [else]<drop/>   * [if oa.name.contains('support')]support: [p.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/]   [else]<drop/>   * [if oa.name.contains('condition')][if (not p.getValue(st, oa.name).oclIsUndefined())]condition:[p.getValue(st, oa.name).oclAsType(String).clean()/][else] <drop/> [/if]   [else]<drop/>   * [if oa.name.contains('passedByRef')] [if (not p.getValue(st, oa.name).oclIsUndefined())][p.getValue(st, oa.name).oclAsType(Boolean)/][else] undefined [/if]   [else]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/for]<drop/>  [/for]<drop/> | [for (c:Comment | p.ownedComment)] <drop/>  [c.\_body.clean()/]  [/for] |

[/for]<drop/>

</table><drop/>

[else][/if]<drop/>

[else][/if]<drop/>

[/for]<drop/>

## Enumerationen

[for (dt:DataType | self.eAllContents(DataType)->sortedBy(name))]<drop/>

[if dt.oclIsTypeOf(Enumeration)]<drop/>

### Enumeration: [dt.name/]

[for (co:Comment | dt.ownedComment)]<drop/>

<dropEmpty>[co.\_body.clean()/]</dropEmpty>

[/for]<drop/>

Contains Enumeration Literals:

[for (e:EnumerationLiteral|dt.oclAsType(Enumeration).ownedLiteral)]<drop/>

* [e.name/]:
  + [for (co:Comment | e.ownedComment)]<drop/>
  + <dropEmpty>[co.\_body.clean()/]
  + </dropEmpty>[/for]<drop/>

[/for]<drop/>

[else] [/if]<drop/>

[/for]<drop/>

## Primitives

[for (dt:DataType | self.eAllContents(DataType)->sortedBy(name))]<drop/>

[if dt.oclIsTypeOf(PrimitiveType)]<drop/>

### Primitiv: [dt.name/]

[for (co:Comment | dt.ownedComment)]<drop/>

<dropEmpty>[co.\_body.clean()/]</dropEmpty>

[/for]<drop/>

[else] [/if]<drop/>

[/for]<drop/>

## Interfaces

[for (it:Interface | self.eAllContents(Interface)->sortedBy(name))]<drop/>

### Interface: [it.name/]

[for (co:Comment | it.ownedComment)]<drop/>

<dropEmpty>[co.\_body.clean()/]</dropEmpty>

[/for]<drop/>

Applied stereotypes:

[for (st:Stereotype | it.getAppliedStereotypes())]<drop/>

* [st.name/]

[for (oa:Property|st.ownedAttribute)]<drop/>

* [if (not oa.name.contains('base'))][oa.name/]: [if (not it.getValue(st, oa.name).oclIsUndefined())][if oa.name.contains('condition')][it.getValue(st, oa.name).oclAsType(String)/] [else][it.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/][/if][else]<drop/>[/if]

[/if] <drop/>

[/for]<drop/>

[/for]<drop/>

[/for]<drop/>

## Signals

[for (si:Signal | self.eAllContents(Signal)->sortedBy(name))]<drop/>

### Signal: [si.name/]

[for (co:Comment | si.ownedComment)]<drop/>

<dropEmpty>[co.\_body.clean()/]</dropEmpty>

[/for]<drop/>

Applied stereotypes:

[for (st:Stereotype | si.getAppliedStereotypes())]<drop/>

* [st.name/]

[for (oa:Property|st.ownedAttribute)]<drop/>

* [if (not oa.name.contains('base'))][oa.name/]: [if (not si.getValue(st, oa.name).oclIsUndefined())][if oa.name.contains('condition')][si.getValue(st, oa.name).oclAsType(String)/] [else][si.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/][/if][else]<drop/>[/if]

[/if] <drop/>

[/for]<drop/>

[/for]<drop/>

<table><drop/>

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attribute Name** | **Type** | **Mult.** | **Stereotypes** | **Description** |

[for (p:Property|si.allAttributes())]<drop/>

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| [p.name/] | [p.type.name/] | [if(p.lower=p.upper)]1[else][p.lower/]..[if(p.upper=-1)]\*[else][p.upper/][/if][/if] | [for (st:Stereotype | p.getAppliedStereotypes())]<drop/>  [st.name/]  [for(oa:Property|st.ownedAttribute)]<drop/>   * [if oa.name.contains('attribute')]AVC: [p.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/]   [else]<drop/>   * [if oa.name.contains('isInvariant')]isInvariant: [p.getValue(st, oa.name).oclAsType(Boolean)/]   [else]<drop/>   * [if oa.name.contains('value')]valueRange: [if (not p.getValue(st, oa.name).oclIsUndefined())][p.getValue(st, oa.name).oclAsType(String).clean()/][else] no range constraint [/if]   [else]<drop/>   * [if oa.name.contains('support')]support: [p.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/]   [else]<drop/>   * [if oa.name.contains('condition')][if (not p.getValue(st, oa.name).oclIsUndefined())]condition:[p.getValue(st, oa.name).oclAsType(String).clean()/][else] <drop/> [/if]   [else]<drop/>   * [if oa.name.contains('passedByRef')] [if (not p.getValue(st, oa.name).oclIsUndefined())][p.getValue(st, oa.name).oclAsType(Boolean)/][else] undefined [/if]   [else]<drop/>   * [if oa.name.contains('reference')][if (not p.getValue(st, oa.name).oclIsUndefined())]reference:[p.getValue(st, oa.name).oclAsType(String).clean()/][else] <drop/> [/if]   [else]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/for]<drop/>  [/for]<drop/> | [for (c:Comment | p.ownedComment)] <drop/>  [c.\_body.clean()/]  [/for] |

[/for]<drop/>

</table><drop/>

[/for]<drop/>

## Operations

[for (op:Operation | self.eAllContents(Operation)->sortedBy(name))]<drop/>

### Operation: [op.name/]

[for (co:Comment | op.ownedComment)]<drop/>

<dropEmpty>[co.\_body.clean()/]</dropEmpty>

[/for]<drop/>

Applied stereotypes:

[for (st:Stereotype | op.getAppliedStereotypes())]<drop/>

* [st.name/]

[for (oa:Property|st.ownedAttribute)]<drop/>

* [if (not oa.name.contains('base'))][oa.name/]:[if (not op.getValue(st, oa.name).oclIsUndefined())][if oa.name.contains('isOperation')][op.getValue(st, oa.name).oclAsType(Boolean)/][else][if oa.name.contains('condition')][op.getValue(st, oa.name).oclAsType(String)/][else][op.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/][/if][/if]

[/if]<drop/>

[/if]<drop/>

[/for]<drop/>

[/for]<drop/>

[if op.ownedParameter->notEmpty()]<drop/>

<table><drop/>

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Parameter Name** | **Type** | **Direction** | **Multi.** | **Description** |

[for (p:Parameter|op.ownedParameter)]<drop/>

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| [p.name/] | [p.type.name/] | [p.direction/] | [if(p.lower=p.upper)]1[else][p.lower/]..[if(p.upper=-1)]\*[else][p.upper/][/if][/if] | [for (c:Comment | p.ownedComment)] <drop/>  [c.\_body.clean()/]  [/for] |

[/for]<drop/>

</table><drop/>

[else][/if]<drop/>

[/for]<drop/>

</gendoc><drop/>