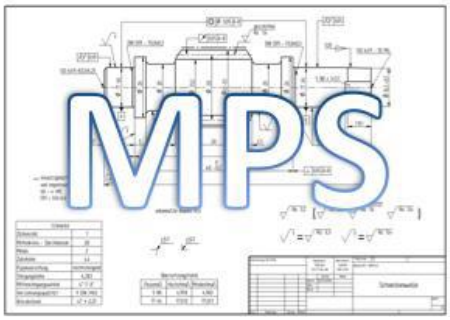


HAW Manufacturing Planning System

Department Informatik

Berliner Tor 7

20099 Hamburg

Designentscheidungen

Version: 5.0
Status: In Arbeit
Stand: 10.06.2014

Zusammenfassung

Dieses Dokument beschreibt Designentscheidungen des HAW- Manufacturing Planning System.

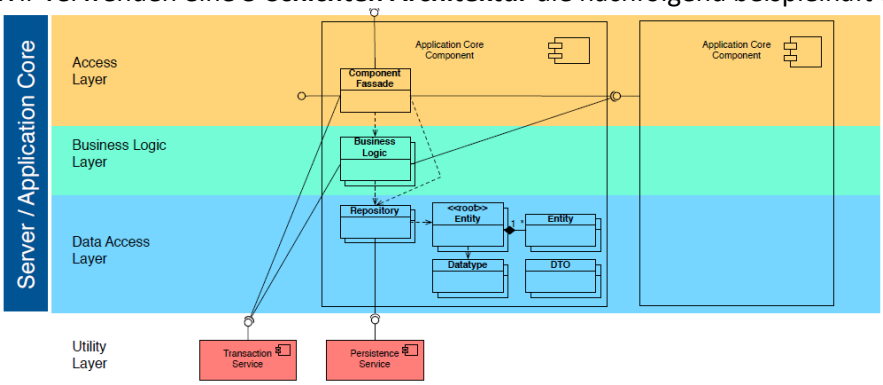
Historie

Version	Status	Datum	Autor(en)
1.0	Veröffentlicht	02.04.2014	Florian Kletz
2.0	Veröffentlicht	28.04.2014	Florian Kletz
3.0	Veröffentlicht	30.04.2014	Florian Kletz
4.0	Veröffentlicht	19.05.2014	Florian Kletz
5.0	Veröffentlicht	10.06.2014	Florian Kletz

Inhalt

Architekturentscheidungen

- Das MPS wird in **Java** Code implementiert.
- Als Persistenz Framework und object-relational mapping Tool wird **Hibernate** eingesetzt.
- Als Datenbank wird eine Zentral genutzte **MySQL** Datenbank verwendet.
- Zur Kommunikation mit Externen System wird eine noch nicht näher spezifiziertes **Message Queue** Tool verwendet.
- Zum Testen des MPS wird das **JUnit** Framework.
- Für Komponenten Tests benutzen wir das **JMock** Framework um nicht zur Verfügung stehende Komponenten nachahmen zu können.
- Wir verwenden eine **3-Schichten Architektur** die nachfolgend beispielhaft dargestellt ist:



- Aufgrund unklarer Aufgabenstellung wird angenommen, das die **Down**-/ **Uptime** (und von uns zusätzlich **Idletime**) als Datum/Uhrzeit gesetzt wird, wenn sich der Status letztmalig geändert hat.
- Die **Heartbeat** Kommunikation zwischen dem **MPS Core** und dem **MPS Monitor** wird über UDP realisiert, da UDP einen geringen Overhead hat und keinen Verbindungsaufbau benötigt.
- Die Kommunikation zwischen den **MPS Core** und dem **MPS Dispatcher** (bidirektional) sowie zwischen dem **MPS Dispatcher** und dem **MPS Monitor** (bidirektional) wird über **TCP** realisiert, da hierfür eine zuverlässige Übertragung der Pakete wichtig ist.
- Die Kommunikation zwischen den **MPS Dispatcher** und **MPS Client *(Server)*** sowie zwischen dem **MPS Monitor** und dem **MPS Dashboard *(Server)*** wird aufgrund geringer Kopplung mittels SOAP realisiert.
- Die Clients die die **GUIs** nutzen stellen eine Verbindung über **HTTP und Websockets** her.
- Für die **JSON** Kommunikation verwenden wir die Bibliothek JSON-Simple.

Annahmen

- Es ist keine Migration von etwaigen Stammdaten nötig um das MPS betreiben zu können.
- Die Firma die das MPS einsetzt, verfügt über den „perfekte“ Lieferanten/Spediteur, d.h. alles was wir bestellen wird unmittelbar und vorrausschauend geliefert und oder abgeholt, sodass keine Bestellungen durch das MPS übermittelt werden müssen. Lagerhaltung ist daher auch überflüssig.
- Die Fertigung liegt außerhalb des Scopes des MPS Projekts und wird daher durch das MPS nur angestoßen, jedoch nicht verwaltet oder gesteuert.
- Fertigungspläne für alle komplexen Bauteile sind allgemein bekannt, sodass sie nicht erstellt oder bereitgestellt werden müssen.

Interne Kommunikation im MPS

JSON Objects zur internen Kommunikation

Heartbeat JSON:

```
{
  "Host": "xxx",
  "Port": "yyy",
  "Systemload": "zzz"
  "Memeory_avail": "aaa"
}
```

createAngebot

```
{
  "Command": "createAngebot",
  "kundenNr": 123,
  "bauteilNr": 123
}

{
  "response": true,
}
```

acceptAngebot

```
{
  "Command": "acceptAngebot",
  "angebotNr": 123,
}

{
  "response": true,
}
```

getAllAngebote

```
{
  "Command": "getAllAngebote",
}


[{
  "angebotNr": 123,
  "gueltigAb": "datum",
  "gueltigBis": "datum",
  "preis": 123.00,
  "status": "status",
  "bauteil": "name",
}]
```

getAllAuftraege

```
{
  "Command": "getAllAuftraege",
}

[{
  "auftragNr": 123,
  "istAbgeschlossen": "enum",
  "beauftragtAm": "datum",
}]
```

getAllBauteile

```
{
  "Command": "getAllBauteile",
}


[{
  "bauteilNr": 123,
  "name": "name",
}]
```

Kommunikation mit Externen Systemen

Spedition

Die Kommunikation mit den Externen Spedition erfolgt über nachfolgend Spezifikation per REST.

SpeditionAdapter > Externe Spedition

Wenn das Bauteil fertig gebaut wurde, wird die Lieferung angestoßen und ein JSON Object an die Spedition geschickt.

```
{
  "transportauftragNr": 123,
  "ausgangsDatum": "datum",
  "lieferungErfolgt": true,
  "transportdienstleister": "name"
}
```

Externe Spedition > SpeditionAdapter

Wenn die Lieferung fertig ist, wird ein JSON Object mit `accept` und der Transportnummer an das MPS geschickt.

Transportauftrag wurde ausgeliefert:

```
{
  "accepted": 123,
}
```

Transportauftrag wurde abgelehnt

```
{
  "rejected": 123,
}
```

Bank

Die Kommunikation für die Externe Bank erfolgt über nachfolgend Spezifikation per MessageMQ

BankAdapter > Externe Bank (HAPSAA)

```
{
  "rechnungsNummer": 123,
  "betrag": 123,
}
```

Externe Bank (HAPSAA) > BankAdapter

Keine Rückantwort