Rendszerintegráció és -felügyelet laboratórium (VIMIM309)

Modell alapú eszközintegráció elosztott környezetben (SDE)

Mérési segédlet

Készítette: Polgár Balázs

Utolsó módosítás: 2011. március 31.

Verzió: 1.0

Budapesti Műszaki és Gazdaságtudományi Egyetem
Méréstechnika és Információs Rendszerek Tanszék

1 Bevezető

A labor során a méréseket végző hallgató a gyakorlatban is megismerkedik a rendszerintegráció és rendszerfelügyelet során használatos módszerekkel és eszközökkel. Végigköveti egy elosztott alkalmazás megvalósításának és felügyeletének legfontosabb lépéseit, ipari környezetben használt integrációs köztes réteg (middleware) technológiák és felügyeleti eszközök használatával. A mérések a következő témakörökhöz kapcsolódnak:

1. Munkafolyamatok megvalósítása Java nyelven
2. Megbízható üzenetküldés IBM WebSphere MQ alapon
3. Kommunikáció JMS és JMX technológia segítségével
4. OSGi szolgáltatások fejlesztése
5. Modell alapú eszközintegráció elosztott környezetben (SDE)
6. Szabályalapú üzleti logika
7. Üzleti folyamatok felügyelete

A jelen mérés során a hallgatók megismerkedhetnek a munkafolyamatok modell alapú tervezésével és felügyeletével. A Service Development Environment (SDE) a szolgáltatásorientált architektúra (SOA) Eclipse/OSGi környezetben való implementációja. A korábbi méréseken megtervezett és OSGi kötegekként megvalósított szolgáltatásokat kell a mérés során adaptálni az SDE környezethez, elosztottan – azaz több OSGi konténerbe – telepíteni, a szolgáltatásokból összeállított munkafolyamatot modellezni és végül a munkafolyamatot végrehajtani és a lefutást a monitorozó felületen nyomonkövetni.

2 A mérés célja

A korábbi mérések során bemutatásra kerültek olyan technológiák, melyek több alkalmazás (rendszer) integrációja során támogatást tudnak nyújtani az alkalmazások közötti kommunikációhoz, ill. a komponensek menedzseléséhez. Míg az első mérés Java környezetben mutatott példát egy alkalmazáson belüli elosztottság megvalósítására szálak felhasználásával, az MQ és JMS technológiák erősen elosztott, heterogén környezetben támogatják az alkalmazások integrációját a köztük lévő kommunikáció szabványos csatornán keresztüli lebonyolításával. Ebben az esetben a különböző technológiával elkészített, ill. különböző nyelven megírt alkalmazások együttműködésének biztosítása a kommunikációs protokoll szabványosításával, és a kommunikációs csatorna különböző platformokon való elérhetőségének biztosításával történt. Itt az egyes lépések végrehajtását az adatok továbbadása váltotta ki.

Az egyes lépések OSGi szolgáltatásként való definiálása az előzőekben említett heterogén alkalmazások integrációját azáltal teszi lehetővé, hogy egy adott környezetben (Java) nyújt lehetőséget az egyes alkalmazások által nyújtott szolgáltatások megosztására. Amennyiben más környezetben készült alkalmazás szolgáltatásait szeretnénk használni, akkor ehhez Java wrapper osztályt kell írni. Egy alkalmazás funkcionalitásának OSGi szolgáltatásként való definiálása egy egyszerű API definiálásához képest annyival nyújt többet, hogy lehetővé teszi a komponensek dinamikusan változtatható integrációját: a szolgáltatás interfészének és implementációjának szétválasztásával az implementációs kódok dinamikusan kicserélhetők egy adott rendszerben. Az OSGi szabvány azonban a szolgáltatások más komponensek számára való elérhetővé tételén túl azok felhasználására már nem tartalmaz útmutatást (a konkrét hívási mód definiálásán kívül).

A szolgáltatásorientált architektúra (SOA) ezen túlmegy: azon túl, hogy a komponensek funkcionalitásának szolgáltatásokként való definiálását írja elő, az integrált rendszer elkészítését a szolgáltatásokból összeállított magas szintű folyamatok definiálásával és menedzselésével javasolja. Itt a dinamizmus már nemcsak az egyes szolgáltatások implementációjának kicserélhetőségére, de a folyamatok összeállítására is vonatkozik.

A Service Development Environment (SDE) a fejlesztési folyamat során használt eszközök SOA alapokon történő integrációját tűzte ki célul. A szolgáltatások deklaratív módon definiálhatók az OSGi deklaratív szolgáltatásaihoz hasonlóan, de az Eclipse környezet kiterjesztési pont mechanizmusát használva[1]. Az SDE ezen túl

- támogatást nyújt a szolgáltatások közötti adatcserére elkülönített futásidejű tárhely biztosításával (SDE Blackboard),
- felhasználói felülettel rendelkezik a regisztrált eszközök megjelenítésére és a szolgáltatások meghívására,
- lehetőséget nyújt a szolgáltatások folyamatba fűzésére egy szkript nyelv segítségével.

Amennyiben biztonságkritikus rendszerek fejlesztése során használt eszközláncok futtatását is szeretnénk támogatni, akkor nem elég, hogy a folyamatot futtatni tudjuk, de annak eredményét rögzíteni is kell a szabványok által előírt módon, hogy a rendszer tanúsítása (biztonságigazolása) során ezeket fel tudjuk használni. A tanszéken ezért kiterjesztettük az SDE keretrendszert olyan folyamat-végrehajtó motor integrálásával, mely támogatja az egyes lefutások eredményeinek perzisztálását, valamint egy általános grafikus folyamatmodellező komponenssel is.

A jelen mérés azt mutatja be, hogy deklaratívan definiált szolgáltatásokat felhasználva hogyan lehet magas szintű folyamatmodellekkel hatékonyan támogatni integrált alkalmazások elkészítését, valamint hogyan használhatók fel a tervezés során elkészített modellek a telepített rendszer működtetése során.

3 Eszközintegrációs keretrendszer

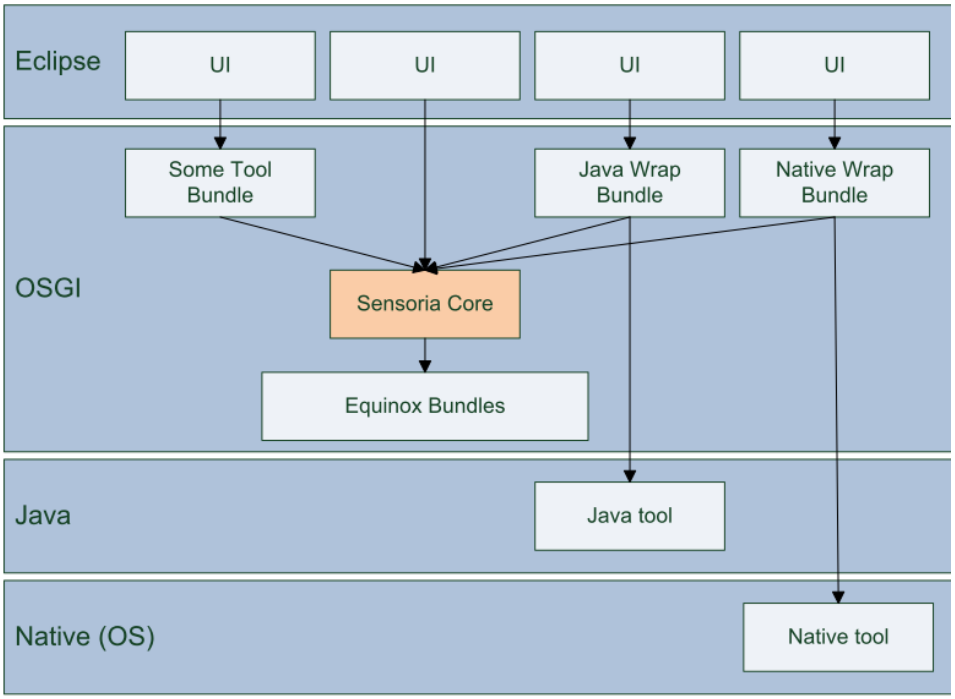
3.1 Service Development Environment

A Service Development Environment (SDE) a SENSORIA EU FP6-os projekt keretén belül került kifejlesztésre különböző fejlesztő és analízis eszközök szolgáltatásorientált integrációjára. A fejlesztőkörnyezet az Eclipse platformra épül, telepíteni az alábbi update site-ról lehet: http://svn.pst.ifi.lmu.de/update/sde. Bővebb információ pedig a http://svn.pst.ifi.lmu.de/trac/sde oldalon érhető el. Az SDE Tutorial részletes útmutatót tartalmaz a környezet használatáról, ill. az eszközök csatolásáról; a következőkben ennek egy rövid összefoglalója következik.

3.1.1 Architektúra

Ahogy már említésre került, az SDE támogatja különböző típusú eszközök (OSGi köteg, egyszerű Java alkalmazás vagy natív OS eszköz) integrációját OSGi kötegekbe csomagolt wrapper osztályoknak a központi komponenshez (Sensoria Core) való csatolásával (1. ábra). Minden egyes komponenshez kapcsolódhat felhasználói felület is; itt beállíthatók például a szolgáltatás futtatásához szükséges paraméterek, de lehetőség van a futás során további információk megadására is (pl. egy platformfüggetlen modell adott platformra való leképzése során megadható a különböző alkalmazáskomponensek hardver csomópontokhoz való hozzárendelése).

[1] Az OSGi deklaratív szolgáltatások az Eclipse 3.5-ös verziójával jelentek meg (2009), míg az SDE fejlesztése a SENSORIA EU FP6-os projekttel már 2005-ben elkezdődött.



1. ábra SDE áttekintő architektúra

3.1.1.1 Helyi eszköztár

Az SDE környezetben egy központi komponens (*Core*) menedzseli az elérhető eszközöket. Ez egyfelől definiál egy kiterjesztési pontot, amihez kapcsolódnak az eszközök (2. ábra), s ez alapján készíti el az elérhető eszközök könyvtárat. Másfelől rendelkezik egy API-val is, amin keresztül lekérhetők és végrehajthatók az egyes eszközök és az általuk definiált szolgáltatások.

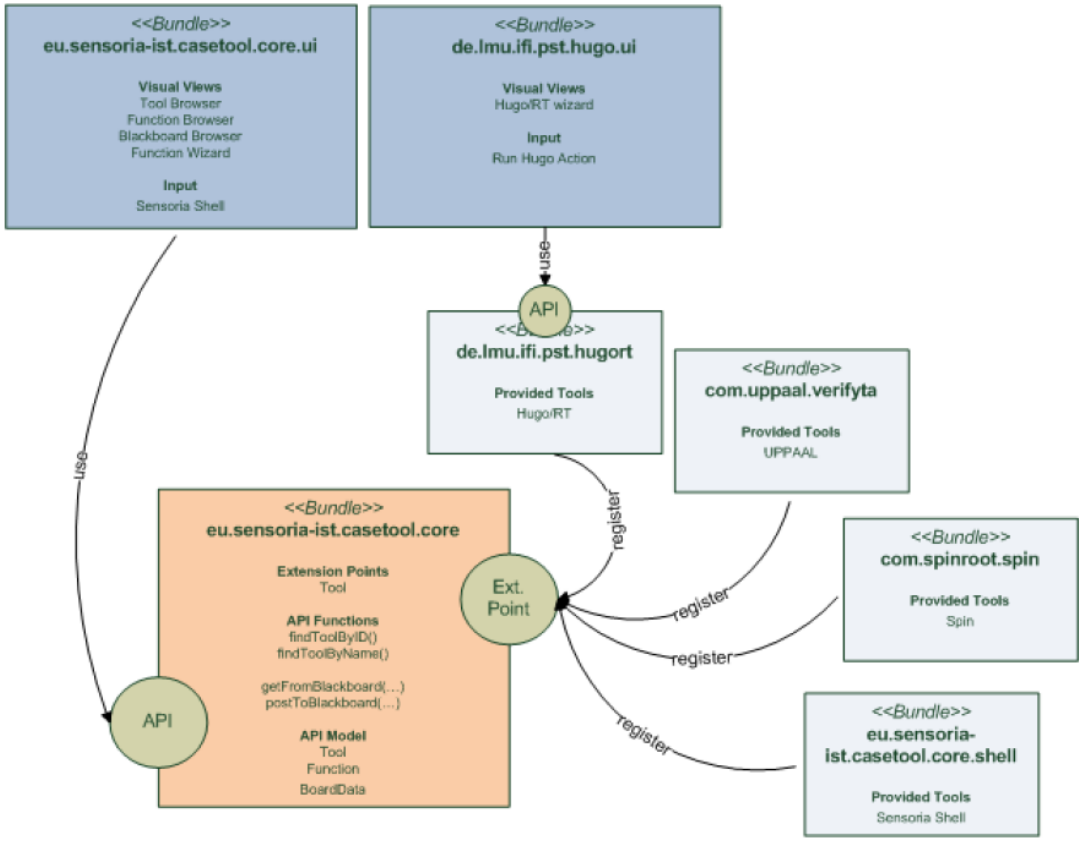
A központi komponens API-jában található függvények is definiálva lettek SDE szolgáltatásokként (pl. *FindToolByID, FindToolByName, postToBoard*), azaz ezek is elérhetők az eszközkönyvtárban: az *SDE Core Service* nevű eszköz tartalmazza ezeket. Részletes lista leírással, paraméterekkel, visszatérési értékekkel megtalálható az eszköz szolgáltatáshívási felületén (ld. 3.1.4. fejezet).

3.1.1.2 Távoli eszköztárak kezelése

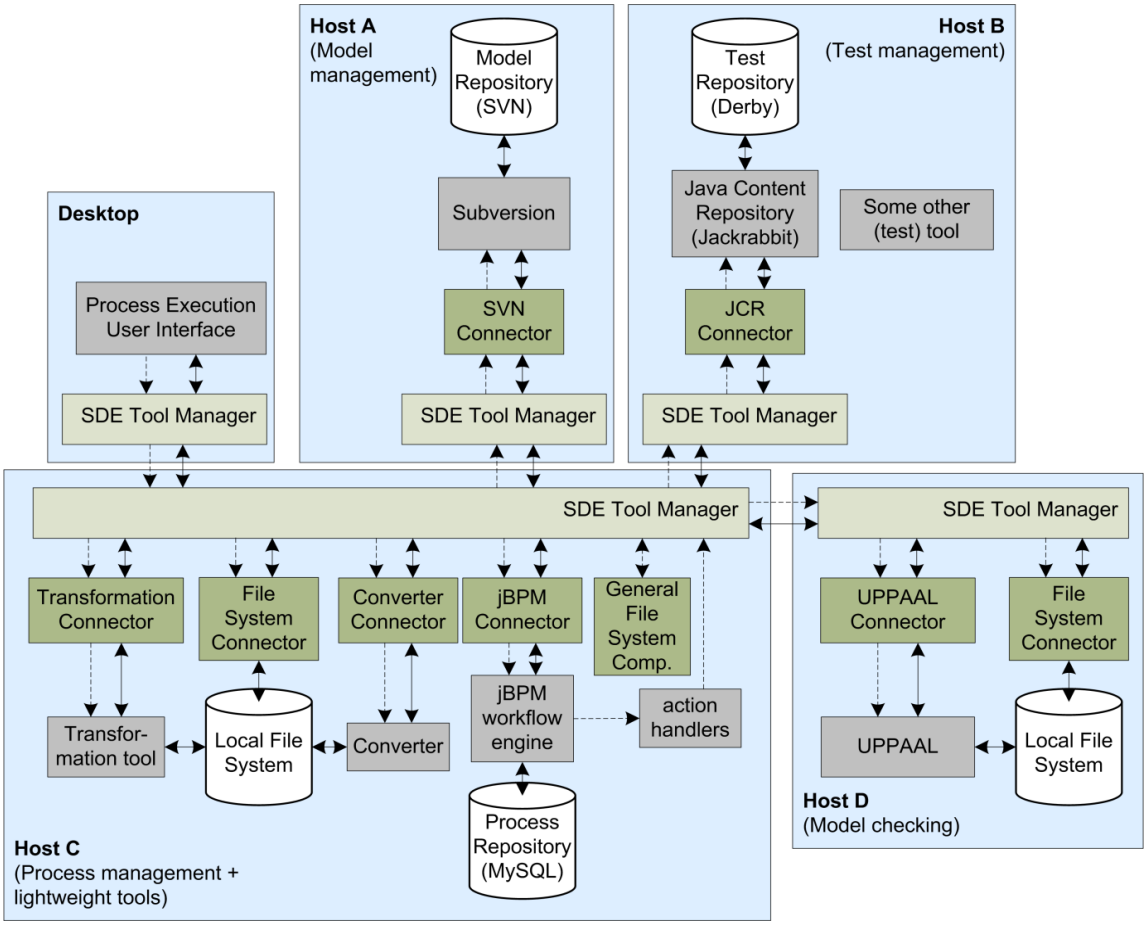
Az SDE keretrendszer lehetőséget biztosít másik OSGi konténerbe telepített eszközök elérésére, mely konténer lehet akár másik gépen is. Ehhez a következők szükségesek:

- Mindegyik OSGi konténerben el kell indítani az SDE komponenst.
- A távoli SDE rendszerben el kell indítani a távoli szolgáltatásszervert: *Remote service server* eszköz *Start* szolgáltatása. Ez meghívható például az eszköz szolgáltatáshívási felületéről (ld. 3.1.4. fejezet). Az IP és a port beállítható a virtuális gép *eu.sensoria_ist.casetool.remote.ip*, ill. *eu.sensoria_ist.casetool.remote.port* paraméterében.
- A helyi SDE rendszerben kapcsolódni kell a távoli SDE-hez a *SDE Core Registry Service* eszköz *addRemoteCore(String locationURI)* szolgáltatásával. A *locationURI*-t "*r-osgi://ip:port*" formában kell megadni. Ettől kezdve azok a távoli SDE-be telepített eszközök is elérhetővé válnak, melyeknek minden szolgáltatása sorosítható paraméterekkel rendelkezik.

A 3. ábra egy példát mutat az SDE keretrendszer és a különböző eszközök, adattárak, folyamatvégrehajtó motor elosztott környezetben való telepítésére.



2. ábra SDE technikai architektúra



3. ábra Példa: különböző eszközök és az SDE keretrendszer telepítése elosztott környezetben

3.1.2 Eszközök kezelése

3.1.2.1 Eszközök csatolása

Eszközöket a rendszerhez csatolni az OSGi szolgáltatásoknál látotthoz hasonlóan rendkívül egyszerűen, deklaratív módon lehet, de ehhez az Eclipse kiterjesztési pont mechanizmusát kell használni. Miután létrehoztunk egy plug-in projektet, és abban egy *ISensoriaTool* interfészt megvalósító osztályt vagy ezt kiterjesztő interfészt az eszköznek, és kijelöltük a szolgáltatáshívási pontokat jelentő függvényeket, ezeket az *SDE Core* által definiált *eu.sensoria_ist.casetool.core.tool* kiterjesztési pont megvalósításával tudjuk elérhetővé tenni a keretrendszer számára.

Például a jBPM folyamatvégrehajtó eszköz folyamatpéldányt végrehajtó szolgáltatását az alábbi módon kell megadni:

```
<extension point="eu.sensoria_ist.casetool.core.tool">
   <tool
      id="hu.bme.mit.toolintegration.workflow.jbpm.IJbpmExecutorTool"
      name="jBPM Executor Tool"
      description="A tool for running jBPM processes, that can access the Core
Registry"
      class="hu.bme.mit.toolintegration.workflow.jbpm.JbpmExecutor">
      <category
         name="Workflow management">
      </category>
      <function
         name="executeProcessInstance"
         returns="void"
         returnsDescription="(no description)"
         description="Executes a process instance.">
         <parameter
              name="processID"
              description="The processInstanceID"
              type="java.lang.String">
         </parameter>
         <parameter
              name="variableMap"
              description="VariableMap(key: varName, value: varValue as a string)"
              type="java.util.HashMap">
         </parameter>
      </function>
      <function ...
      </function>
   </tool>
</extension>
```

Az SDE keretrendszer nyújt egy olyan kényelmi szolgáltatást, hogy ha az eszközosztályunkat, ill. interfészünket és a szolgáltatáshívási függvényeinket megfelelő módon felcímkézzük, akkor ez a kiterjesztési pont definíció automatikusan generálható. A fenti példánál ez az alábbiakat jelenti:

```
@SensoriaTool(categories = "Workflow management",
     description = "A tool for running jBPM processes, that can access ...",
     name = "jBPM Executor Tool")
public interface IJbpmExecutorTool extends ISensoriaTool {

   @SensoriaToolFunction(description = "Executes a process instance.")
   public void executeProcessInstance(
        @SensoriaToolFunctionParameter(description = "The processInstanceID")
        String processID,
        @SensoriaToolFunctionParameter(description = "VariableMap(key: varName,
value: varValue as a string)")
        HashMap<String,String> variableMap
   ) throws Exception;

}
```

Ezután az interfész fájlon a kontextus menüben elérhető a *Convert to Sensoria tool* parancs, ami egy ablakba legenerálja a kiterjesztési pont definícióját. Ezt át kell másolni a plugin.xml fájlba és a *class*-t kézzel (!) be kell állítani arra az osztályra, ami ennek az interfésznek a megvalósítását adja.

Ahhoz, hogy az eszközünk forduljon és futásidőben elérhető legyen, két dolog kell még:

- a függőségek közé fel kell venni az *eu.sensoria_ist.casetool.core* OSGi köteget,
- a megfelelő csomagokat exportálni kell, hogy az *SDE Core* számára elérhetőek legyenek.

3.1.2.2 Eszközök meghívása

Az eszközkönyvtárba regisztrált eszközöket többféleképpen is meg lehet hívni:

- *Felhasználói felületről*

  A keretrendszer lehetőséget biztosít a felületén az eszközök tallózására és a kiválasztott eszköz szolgáltatáshívási felületén az általa nyújtott szolgáltatások meghívására a paraméterek megadásával együtt (ezeket egy külön ablakban be lehet gépelni vagy tallózással kiválasztani).

  Részletes leírás a 3.1.4-es fejezetben található.

- *Shell utasítással*

  Egy shell ablakban a szolgáltatások közvetlenül is meghívhatók. Ehhez a központi komponenstől előbb el kell kérni egy referenciát az eszközre, és utána lehet meghívni valamelyik szolgáltatását a megfelelő paraméterekkel. A központi komponensre az *sCore* változóval lehet hivatkozni:

  *eszköz = sCore.findToolById(*toolID*).getServiceInterface();*

  *eszköz.*szolgáltatás*(*param1*, ...);*

  Például:

  jbpm = sCore.findToolById("hu.bme.mit.toolintegration.workflow.jbpm.IJbpmExecutorTool").getServiceInterface();

  processID = ...;

  params = ...;

  jbpm.executeProcessInstance(processID,params);

  Az utasítások szintaktikáját a JavaScript nyelv adta lehetőségek határozzák meg.

- *Programból*

  Az *SDE Core* biztosít egy reflektív API-t, amin keresztül Java programból is el lehet kérni név vagy ID alapján az eszközöket és futtatni adott nevű szolgáltatásaikat a megfelelő paraméterekkel.

3.1.3 SDE eszközök folyamatba szervezése

Az eszközök által nyújtott szolgáltatásokat tipikusan egy összetettebb folyamat egy lépéseként hívjuk meg. A szolgáltatások folyamatba szervezése során fontos feladat az adatok átadása az egyes lépések között; a folyamatba szervezésre és adatátadásra az SDE keretrendszer az eszközök hívásához hasonlóan többféle lehetőséget is nyújt:

- *Kézi végrehajtás*

  A felhasználói felületről hívva az eszközöket egymás után, ezek láncba fűzhetők. Az adatok átadására a keretrendszer biztosít egy *futásidejű adattárolót* (*SDE Blackboard*), ahová tetszőleges objektumot ki lehet tenni, ill. a szolgáltatások paramétereinek megadásánál ott lévő objektumot fel lehet használni.

- *Szkript alapú folyamatba szervezés*

  Az említett shell utasításokat egy szkript fájlba is be lehet tenni; ekkor az adatáramlás tipikusan a változókon keresztül történik. A fájlnak az *.sscript* kiterjesztést kell adni; ekkor a kontextus

menüben a *Run as / Sensoria shell script* ill. *Sensoria shell script NB* paranccsal futtatható a szkript. Az NB-s verzió nem blokkoló módban futtatja.

- *Folyamatok modellezése*

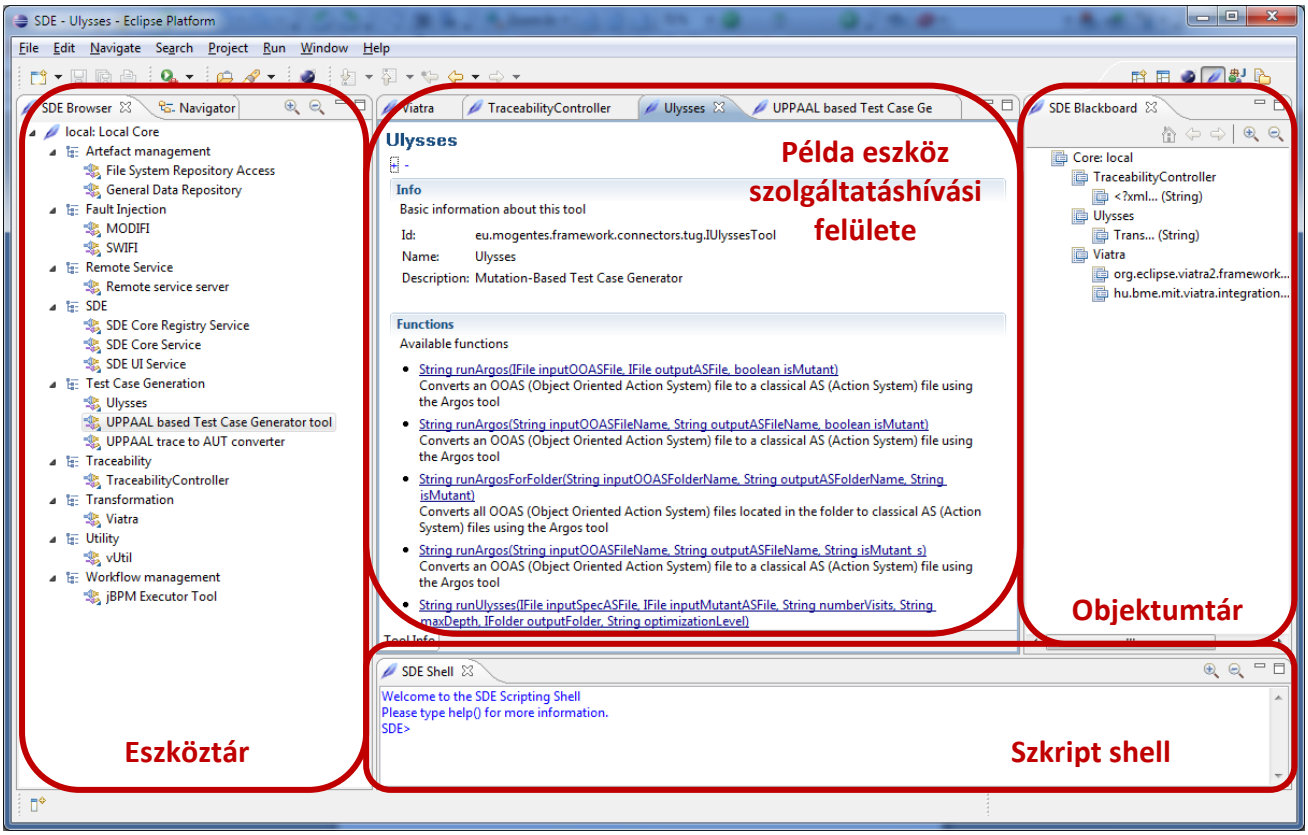
  Az SDE keretrendszer biztosít egy egyszerű grafikus szerkesztőt is a szolgáltatások folyamatba szervezésére, amivel a szkript már modell alapon elkészíthető. Ez azonban csak egyszerű, elágazás és párhuzamos ágak nélküli folyamatokat enged létrehozni. Ezért nem ezt fogjuk használni, hanem egy ehhez hasonló, de kiterjesztett képességekkel rendelkező folyamatszerkesztőt (ld. 3.2. fejezet).

3.1.4 Az SDE felhasználói felülete

Az SDE felhasználói felület három Eclipse-es nézetből áll (4. ábra):

- Eszköztár (*SDE/SDE Browser*)
- Objektumtár (*SDE/SDE Blackboard*)
- Szkript shell (*SDE/SDE Shell*)

Ezek az *SDE* perspektíva betöltésével együtt is megnyithatók.



4. ábra SDE nézetek

Az *eszköztár* tartalmazza az összes regisztrált eszközt, ami az SDE környezetben elérhető. Az eszközök kategóriák szerint csoportosítva jelennek meg. Amennyiben távoli SDE komponens is regisztrálásra került, akkor az itt külön *Core*-ként megjelenik az általa kezelt eszközökkel együtt.

Egy eszközt megnyitva (dupla kattintás) megjelenik az *eszköz szolgáltatáshívási felülete*. Itt található az eszköz által definiált szolgáltatások felsorolása. A szolgáltatások közvetlenül meghívhatók erről a felületről: a szolgáltatás kiválasztása után egy felugró ablakban megjelenik a paraméterek listája, amik egyesével beállíthatók, majd a szolgáltatás elindítható. A paramétereket többféleképpen is meg lehet adni: vagy közvetlenül megadunk egy értéket, vagy a fájlrendszerből választunk ki egy fájlt, vagy pedig az objektumtárból választunk egy megfelelő típusú objektumot.

Az *objektumtár* egy olyan futási idejű tároló, mely a szolgáltatásoknak az előbb említett felületről történő kézi meghívása esetén a szolgáltatások közötti adatcserét teszi lehetővé. Ide kerülnek alapértelmezésben a szolgáltatások visszatérési értékei, de az egyes szolgáltatások programozottan is tudnak ide tenni objektumokat.

A *szkript shell* egy JavaScript alapú környezetet biztosít a szolgáltatások programozott hívására és ezáltal láncba (folyamatba) fűzésére.

3.2 Folyamatszerkesztő

A tanszéken kifejlesztésre került egy általános célú folyamatmodellező eszköz, mely használható az SDE eszközök szolgáltatásainak folyamatba szervezésére is.

3.2.1 Folyamat metamodell

A modellező eszköz egy folyamat metamodellen alapul, mely metamodell konform az UML aktivitásmodelljével[2].

A folyamat csomópontokból áll, melyek lehetnek szolgáltatás-, adat-, ill. vezérlő csomópontok. A szolgáltatás-csomópont jelenít meg értelemszerűen egy szolgáltatáshívást. Ennek lehetnek bemeneti és kimeneti pinjei, melyek a szolgáltatás bemeneti és kimeneti adatait reprezentálják. A pinek mellett adatcsomópont lehet még a folyamat bemeneteit reprezentáló paraméter-csomópont is. Vezérlő csomópontok a kiinduló és végállapotok, elágazó és szinkronizáló csomópontok (*fork*, *join*), valamint a döntési és egyesítő csomópontok (*decision*, *merge*). A szolgáltatás és vezérlő csomópontok között lehet vezérlésáramlást, adat és vezérlő csomópontok között pedig adatáramlást definiálni.

A részletes metamodell a *Függelék*-ben található.

3.2.2 Folyamat modell

A metamodell alapján egy folyamatmodell példány a *New... / Examples/ProcessModel Diagram* varázsló használatával hozható létre.

Szolgáltatás-csomópontok esetén a *Name* tulajdonság tartalmazza a csomópont címkéjét, a *Service ID* a szolgáltatás nevét, a *Tool ID* pedig a szolgáltatást tartalmazó eszköz nevét. Az eszköznév *hoszt#eszközID* alakú, ahol a *hoszt* adja meg, hogy melyik csomóponton fut az alkalmazás, az *eszközID* pedig az eszköznek a kiterjesztési pont definícióban megadott azonosítója. A helyi OSGi konténerben futó eszközök esetén a *hoszt* a *local* kulcsszóval hivatkozható, távoli konténerek esetén pedig azok *ip:port* azonosítójával.

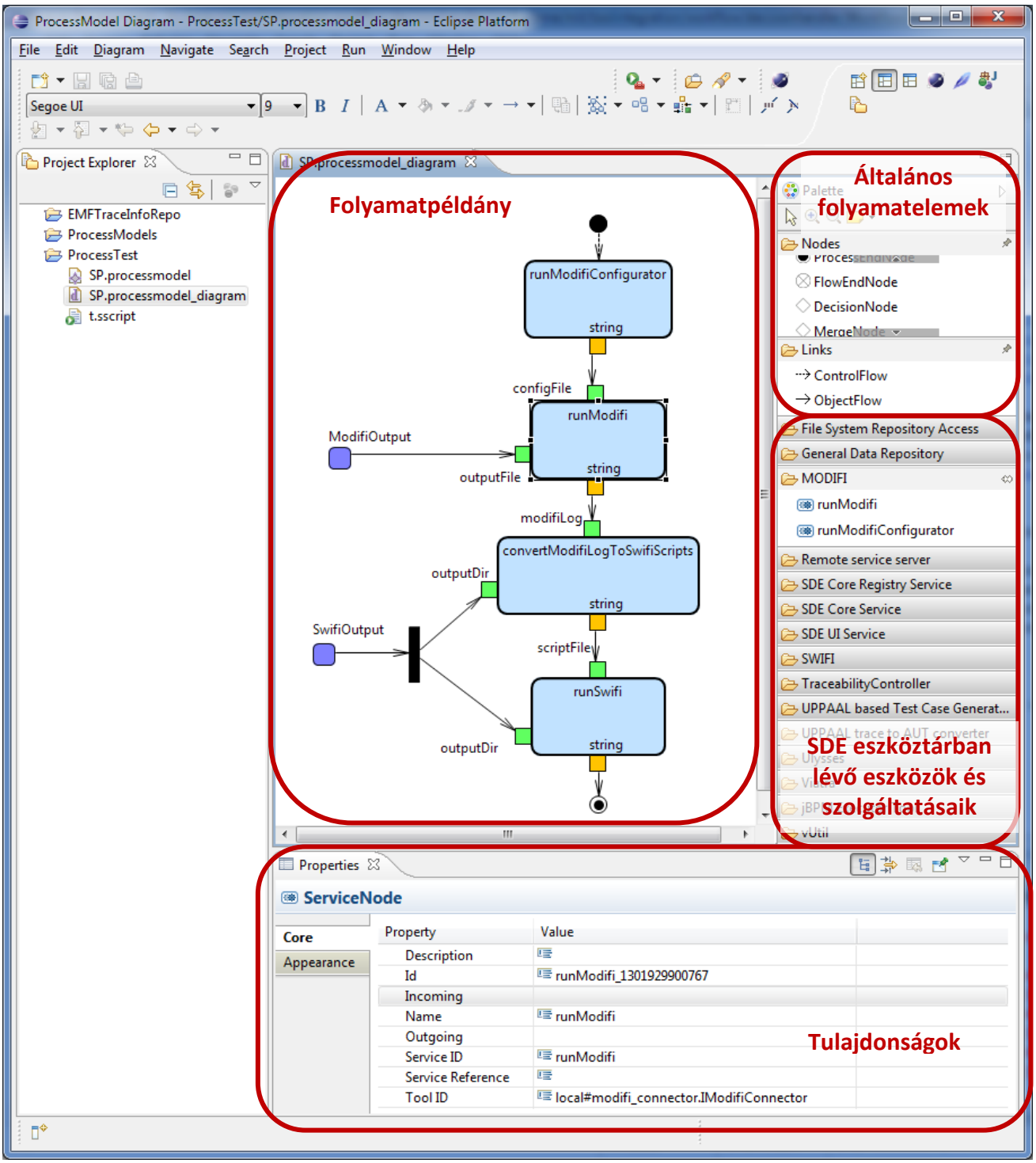
Döntési csomópontok esetén a bemeneti élnek adatáramlásnak kell lennie, és a feltételnek erre az adatra kell vonatkoznia. A feltétel tetszőleges nyelven leírható a nyelv és a feltételkifejezés megadásával. A kiértékelő jelenleg a *javascript* nyelvet támogatja. A kifejezésben a *decisionInput* változóval lehet hivatkozni a bemeneti adatra és a *decisionOutput* szöveges változó értékét kell beállítani valamelyik kimenő él címkéjére. A végrehajtás a kiválasztott kimenő élen fog folytatódni.

3.2.3 Szerkesztő felület

Az előző pont szerint létrehozott *.processmodel_diagram* fájl megnyitásakor a folyamatmodell a grafikus szerkesztőben nyílik meg (5. ábra). A jobb oldali palettán elérhetőek a metamodell által definiált általános elemek, valamint megjelennek az SDE eszköztárban definiált eszközök és szolgáltatásaik is. Amennyiben ez utóbbi elemeket tesszük le a szerkesztőfelületre, automatikusan létrejön a megfelelő tulajdonságokkal rendelkező szolgáltatás-csomópont a paramétereit jelképező pinekkel együtt.

A folyamatszerkesztéshez szükséges nézeteket a *Processmodel Diagram* perspektíva tartalmazza.

[2] Az elkészített folyamat metamodell egész pontosan az UML aktivitásmodelljének a folyamatok modellezéséhez szükséges koncepciókat tartalmazó részhalmazaként lett definiálva.



5. ábra Folyamatszerkesztő felület

3.3 Folyamatvégrehajtás és monitorozás

A folyamatmodellező eszköz mellé a tanszéken kifejlesztésre került egy végrehajtó és monitorozó környezet (*Process Manager*). A megvalósítás során az volt a cél, hogy az elkészített platformfüggetlen folyamatmodellünket különböző létező (*off-the-shelf - OTS*) folyamatvégrehajtó motorok segítségével végre tudjuk hajtani, ugyanakkor ne legyünk kötve egyik, ill. másik implementációhoz. Ezért egy olyan megoldás került kidolgozásra, ami lehetővé teszi különböző motorok integrálását és akár együttes kezelését is.

3.3.1 Process Manager architektúrája

A folyamatvégrehajtó és monitorozó komponens az alábbi részekből áll:

- *Végrehajtó motor*

  Ez egy olyan OTS komponens, ami valamilyen folyamatdefiníció alapján képes végrehajtani egy folyamatot. Egy ilyen komponens integrálásához le kell képezni az általános folyamatdefiníciót a végrehajtó motor által igényelt szintaxisra, meg kell oldani az egyes lépésekben a hivatkozott szolgáltatások hívását és állapotjelentést kell küldeni a végrehajtás állapotáról. A rendszerbe jelenleg a jBPM folyamatvégrehajtó komponens van integrálva.

- *Folyamatkatalógus*

  Ez tartja nyilván az elérhető folyamatdefiníciókat, ill. hogy ezek milyen motorra lettek telepítve és ott milyen példányok lettek létrehozva.

- *Folyamatmenedzser*

  Ez egy általános folyamatkezelő komponens, mely

  - kezeli a folyamatkatalógusban lévő elemeket:
    - folyamatdefiníció hozzáadása, törlése;
    - végrehajtó motorra való telepítés, ill. telepített folyamat törlése;
    - folyamatpéldány létrehozása, törlése;
    - folyamatpéldány paramétereinek beállítása;
  - lehetőséget ad a folyamatok elindítására,
  - figyeli a végrehajtó motorról érkező üzeneteket és ez alapján frissíti a folyamatpéldány állapotát.

3.3.2 Folyamatvégrehajtás menete

A folyamatvégrehajtás az alábbi lépésekből áll:

1. A folyamatmodell (*.processmodel_diagram*) hozzáadása a folyamatkatalógushoz.
2. Folyamatmodell telepítése valamelyik végrehajtó motorra.
3. Folyamatpéldány létrehozása.
4. Folyamatpéldány paramétereinek megadása (ha vannak, ill. ha az alapértelmezettől eltérő értékkel akarjuk futtatni a folyamatot).
5. Folyamatpéldány futtatása és a futtatás menetének nyomonkövetése.

3.3.3 Process Manager felhasználói felülete

A folyamat végrehajtásához és a lefutás monitorozásához szükséges nézeteket a *Process manager* perspektíva tartalmazza (6. ábra). Ezek a következők:

1. *Processes* nézet

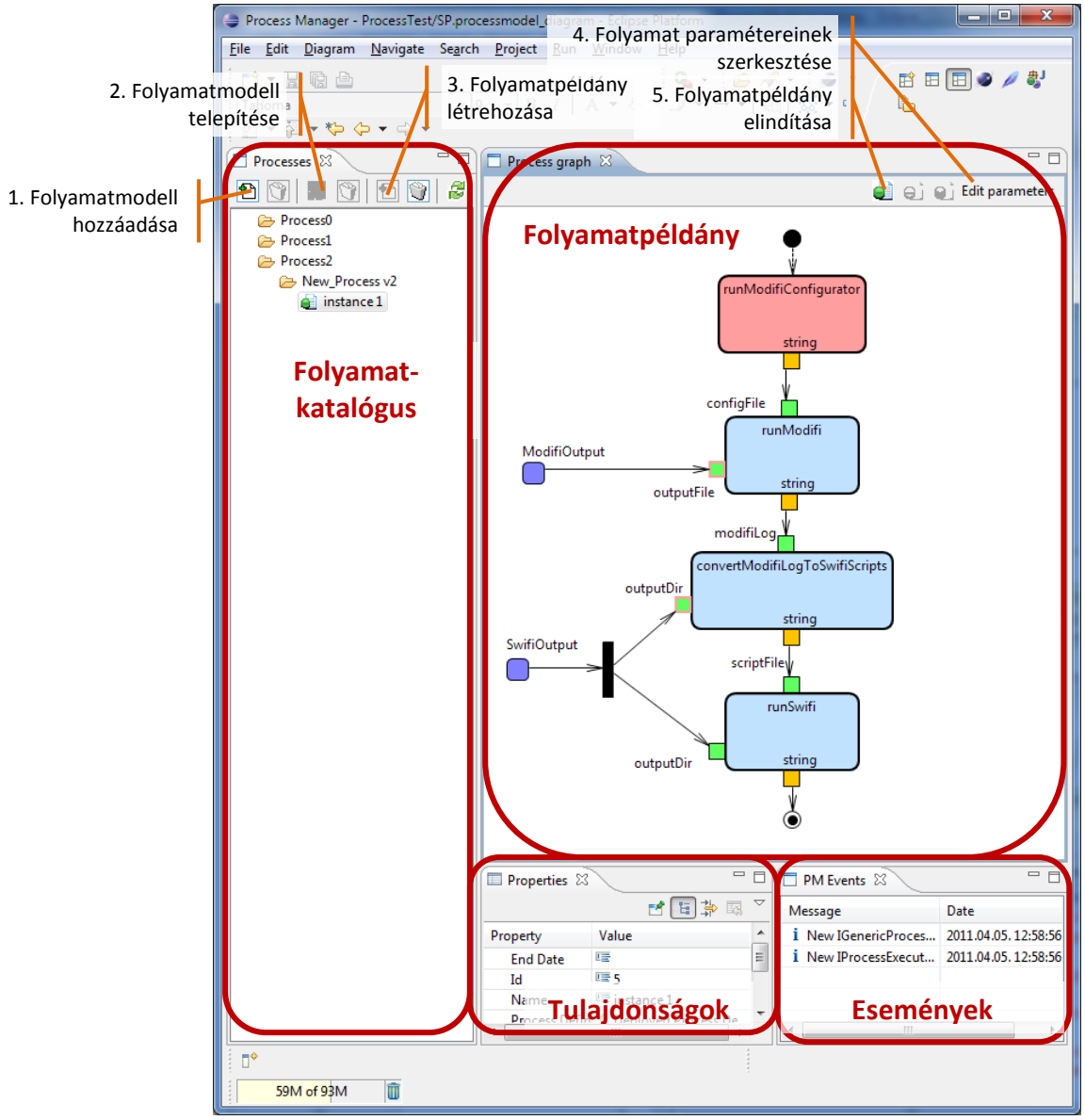
   Ez a nézet tartalmazza a folyamatkatalógusba feltöltött folyamatokat, ezek telepített változatait, ill. a belőlük létrehozott folyamatpéldányokat, valamint itt lehet elvégezni a hozzájuk kapcsolódó műveleteket.

2. *Process graph* nézet

   Ez a nézet a *Processes* nézetben kiválasztott folyamat grafikus reprezentációját tartalmazza (ahogyan az a szerkesztőben létre lett hozva). Folyamatpéldány esetén itt lehet szerkeszteni a paramétereket, elindítani a végrehajtást és nyomonkövetni a lefutást.

3. *PM Events* nézet

   Itt találhatók diagnosztikai üzenetek, melyek a *Process Manager* működésével kapcsolatosak.



6. ábra Process Manager

4 A mérés elvégzése

4.1 Mérési feladatok

A jelen mérés során a hallgatóknak az első mérésre elkészített folyamatot kell implementálniuk, futtatniuk és monitorozniuk a fent leírt környezetben. Ehhez az alábbi részfeladatokat kell elvégezniük:

1. A folyamat egyes lépéseit SDE eszközök szolgáltatásaiként kell implementálni oly módon, hogy a szolgáltatások legalább három különböző eszköz között legyenek szétosztva, és az egyes eszközök legalább két külön OSGi kötegbe kerüljenek. Ehhez praktikusan felhasználható az OSGi mérésen elkészített implementáció.
2. Az eszközöket két különböző OSGi konténerbe kell telepíteni (praktikusan két runtime Eclipse-et kell indítani). Ezek közül az egyiken fog futni a végrehajtó motor, aminek mind a helyi, mind a távoli eszközöket el kell tudnia érni. A hallgatók teszteljék az elkészített eszközöket és a

közöttük lévő adatcserét az SDE felületéről történő kézi hívással és az objektumtár használatával.

3. El kell készíteni a saját folyamat modelljét.
4. A folyamatmodellt telepíteni kell a jBPM folyamatvégrehajtó motorra.
5. A működést demonstrálni kell a folyamat végrehajtásával és a lefutás nyomonkövetésével mind egy, mind pedig több példány elindítása esetén.

A futás során vezérelhető legyen az egyes lépések végrehajtása az üzleti logika részeként (például a szolgáltatás belsejéből a *JOptionPane.showMessageDialog()* függvény meghívásával).

4.2 A mérés kiértékelése

A mérés elfogadásának feltétele a futó folyamat bemutatása és az elvégzett munkáról készített jegyzőkönyv leadása.

A jegyzőkönyv mindenképp tartalmazza a következő elemeket:

- Folyamat magasszintű leírása
- Implementált eszközök és az általuk nyújtott szolgáltatások leírása
- A telepített rendszer felépítése
- A folyamatszerkesztőben elkészített folyamatmodell bemutatása
- A futtatás és nyomonkövetés dokumentálása
- A fejlesztés tapasztalatai, buktatók, elkövetett hibák
- Az elkészített rendszer rövid összefoglaló értékelése

Ezen kívül kerüljön bele a jegyzőkönyvbe minden egyéb olyan részlet is, amit a hallgató fontosnak tart.

A dokumentálásnál a hallgató törekedjen arra, hogy a jegyzőkönyvből mind az elvégzett munka nyomonkövethető legyen, mind pedig az elkészített rendszer kellőképpen megismerhető legyen. Ehhez szükség esetén készítsen ábrákat, komponensdiagramokat, képernyőképeket, illesszen be kódrészleteket, de kerülje a felesleges ismétléseket, hosszú kódokat.

A jegyzőkönyvhöz mellékelni kell az elkészített rendszer forráskódját exportált Eclipse projektekként.

5 Ellenőrző kérdések

1. Mennyiben más, ill. több az SDE megközelítése a szolgáltatások kezelését illetően az OSGi megközelítésénél?
2. Milyen szolgáltatásokat nyújt az SDE keretrendszer?
3. Mi a szolgáltatásorientált architektúra alapelve?
4. Mik a felhasznált folyamatmodell főbb elemei?
5. Hogyan épül fel a Process Manager komponens?
6. Milyen lépések kellenek a folyamatok modell alapú futtatásához?
7. Mik az előnyei a folyamatok modell alapú futtatásának?

Függelék – Folyamat metamodell