

TDK dolgozat

Szerző(k):

Ábrahám Gyula

Pannon Egyetem

Műszaki Informatikai Kar

Villamosmérnöki és Információs Rendszerek Tanszék

Üzleti folyamatok ágens alapú szimulációja adott
időpontban rendelkezésre álló erőforrások ütemezésének
figyelembe vételével

Szerző(k):

Ábrahám Gyula

4. évfolyam

Témavezető(k):

Starkné Dr. Werner Ágnes

tanszékvezető, egyetemi docens

Tartalmi összefoglaló

Az üzleti folyamatok elemzése stratégiai fontosságú lehet az optimális végrehajtás érdekében. Jelen tanulmányban egy olyan megoldás kerül bemutatásra, amely üzleti folyamatok ágens alapú szimulációs vizsgálatát teszi lehetővé. A kidolgozott módszer célja egy olyan végrehajtási stratégia megadása, amely a vizsgált folyamat megbízhatóbb és hatékonyabb lefutását eredményezi. A módszer végrehajtása során egy speciális adattárházban kerülnek tárolásra az üzleti folyamatokat leíró folyamatmodellek és a hozzájuk tartozó erőforrások definíciói. Egy log adatbázis is rendelkezésre áll, amely a folyamatok végrehajtására vonatkozó historikus adatokat, azaz a valós lefutásokhoz tartozó ismérveket rögzíti. A logban tárolt információk elemzéséhez, valamint a logban tárolt adatok és a folyamatmodellek összehasonlításához algoritmusokat fejlesztettünk ki, amelyek az erőforrások kiosztására vonatkozó hasznos információkat szolgáltatnak. A módszerhez tartozóan kidolgoztunk egy olyan ágens szimulációs eljárást is, amelynek felhasználásával a folyamatok erőforrás szükségletének optimális ütemezése megadható. Ily módon a korábbi tapasztalatok felhasználásával egy üzleti folyamat megoldása egyre jobbá és hatékonyabbá tehető, elkerülhetőek a túlvállalások és ezáltal a határidőn túli teljesítések, mérlegelhetőek és hatékonyan kezelhetőek a nem kívánt események, mint például egy erőforrás váratlan meghibásodása.

Kulcsszavak: üzleti folyamat, ágens, erőforrás ütemezés, optimalizálás

Tartalomjegyzék

1. Irodalmi áttekintés	1
2. A komplex feladat bemutatása	4
3. Döntéstámogató alrendszer	9
3.1. Ágens alapú szimulációs modul	9
3.1.1. Előfeldolgozó (Preprocessing)	11
3.1.2. Vezérlő ágens (Controller Agent)	12
3.1.3. A platform	12
3.1.4. Adatbázis felügyelet	13
3.2. Kommunikáció, protokoll, automatizált mechanizmusok	14
3.2.1. Alkalmazott kommunikációs protokollok	14
3.2.2. Kommunikációs modell	14
3.3. Erőforrások kezelése	16
3.4. Folyamatmodellek feldolgozása	18
3.4.1. Munkafolyamat megadása absztrakt formában	18
3.4.2. Munkafolyamatok a döntéstámogató alrendszerben	18
3.5. Ütemezés	22
3.5.1. Általános ütemezési modell	23
3.5.2. Időalapú ütemezés	24
3.5.3. Költségalapú ütemezés	24
3.6. Algoritmusok	25
3.7. Grafikus modul	28
3.7.1. Főablak	28
3.7.2. Erőforrástípusok fül	29
3.7.3. Aktivitások fül	30
3.7.4. Terméktípusok fül	31
3.7.5. Gantt-diagram fül	32
4. Összefoglaló	34

1. Irodalmi áttekintés

A felhő technológia alkalmazása az utóbbi években nyert teret az információs technológiák fejlesztése területén. Gyors terjedésének egyik oka, hogy a felhőben megvalósított rendszerek és szolgáltatások bárhol és bármilyen számítástechnikai eszközről - laptopról, mobilról stb. - elérhetők. A felhő alapú architektúra sokféle rendszer fejlesztéséhez adhat alapot. Gorbenko és társai cikkében [10] az erőforrásütemezés problémáját közelítik meg a felhő alapú rendszerek oldaláról. Mivel az erőforrásütemezés problémája NP-teljes, így a cikk ennek optimális kezelésére keres megoldást. Egy logikai modellt alkalmaz, amelynek segítségével különböző algoritmusok dolgozhatók ki a probléma megoldására. Továbbá ez a modell PSO (Particle Swarm Optimization) algoritmusok tesztelésére is alkalmas, amelyeket munkafolyamatok ütemezéséhez alkalmazhatunk. [3]-ben a felhő alapú rendszerekben történő munkafolyamat ütemezést helyezik előtérbe. Elsősorban olyan algoritmusok keresését tűzték ki célul, amelyek a munkafolyamatok ütemezését megbízhatóvá teszik, s ezáltal számottevően javul a rendszer teljesítménye. A munkafolyamat ütemezésben a cikk három fontos komponenst különböztet meg: maga a munkafolyamat ütemezés, adatáramlás és a hibakezelés. [18]-ben a szerzők a felhő alapú megközelítést használva a munkafolyamatok optimalizálására helyezik a hangsúlyt. Három fontos lépésben teszik ezt: erőforrások felfedezése és allokálása, kiszolgálás minőségi paramétereinek (pl. futási idő, megbízhatóság, költség) meghatározása, valamint kivételkezelés (ha egy adott erőforrás nem elérhető az adott feladathoz). Szintén a felhő alapú megközelítést kihasználva Gao és társai [11] munkájukban a rendszer teljesítményének stabilizálására és növelésére próbálják helyezni a hangsúlyt. Ezt a munkafolyamatok futtatása közben jelentkező váratlan események (hibák) kiküszöbölésével, vagy legalábbis minimalizálásával próbálják elérni. A mi munkánk célja egy olyan felhő technológián alapuló rész szoftver rendszer kifejlesztése, amely lehetőséget nyújt arra, hogy a leghatékonyabban erőforrásokat rendeljünk az egyes munkafeladatokhoz.

Világszerte egyre több vállalat újratervezi az alapvető üzleti folyamatokat, hogy az jövedelmezőbb legyen és az ügyfél-elégedettség is növekedjen. A modellezés és az elemzés két fontos lépés minden folyamat áttekintésénél. Bhaskar és társai [6] egy összefoglalást adnak azokról a szimulációs eszközökről, amelyekkel hatékonyan vizsgálhatóak, dokumentálhatóak és szimulálhatóak az üzleti munkafolyama-

tok. Tumay [17] munkájában szintén az üzleti folyamatok szimulációjára koncentrált. A tanulmányban többek között a modellezési elemek és a modellteljesítmény mérésével foglalkozik. Továbbá helyet kapott az üzleti folyamatok osztályozása, egy egyedi modellezési eljárás, a szimulációs folyamat bemutatása és különböző elemzési megfontolások. [9]-ben egy olyan szimulációs modellel foglalkoznak a szerzők, amely az elvégzendő feladatok és az elosztott emberi erőforrások ütemezését teszi lehetővé ipari folyamatok esetében. A fő pillérek, amelyek mentén a model dolgozik és próbálja minimalizálni az egyes feladatok időbeni költségeit: ingadozás, rövidtávú szolgáltatási igények, amelyek az erőforrások elérhetőségétől függenek, rugalmas feladatvégzés, elmaradt feladatok pótlása. Az optimális végrehajtást a model az egyes feladatokhoz a megfelelő erőforrás hozzárendelésével oldja meg idő és költség alapján. A mi munkánkban is helyet kap a szimuláció, amelynek a segítségével az erőforrás hozzárendelést követően a munkafolyamat lefolyását követhetjük nyomon.

Az operációkutatás egyik fontos és igen széles körben használt területe az ütemezés. Az iparban sok területen előnyösen használható optimális ütemtervek elkészítéséhez vagy erőforrás elosztási feladatok megoldásához. Számos különböző algoritmus látott napvilágot folyamat elemeinek szervezéshez (lásd pl. [14], [7]), amelyek kapcsolatban állnak például a számítógépes hálózatokkal [5], az üzleti folyamatokkal [4], [19] vagy ipari folyamatokkal [16]. Az eredményeket általában Gantt-diagramon ábrázolják, amelyet mi is felhasználunk a döntési javaslat szemléltetésénél. Ezek a módszerek az erőforrásokat általában független komponensként kezelik, nem veszik figyelembe az erőforrások közötti esetleges kapcsolatokat. Csak néhány olyan publikáció található, amely vizsgálja az együttműködés lehetőségét az erőforrások ütemezésében, lásd például [13] vagy [8] munkákat.

A munkafolyamatok hálózaton keresztül történő optimális feldolgozására helyezik a szerzők a hangsúlyt [20]-ben. A problémát a hálózatok irányából közelítik meg, és szűk keresztmetszetnek az esetleges gyenge hálózati erőforrásokat tekintik. A tanulmányban egy olyan rendszert mutatnak be, amely a hálózati erőforrások optimális kihasználásával próbálja a feldolgozást optimalizálni, azaz a hálózat lassú erőforrásait kiszűrni. Ezzel a megközelítéssel a mi munkánkban még nem foglalkoztunk eddig. Bár a mi rendszerünk is hálózaton keresztül kapja az adatokat, a feldolgozás viszont helyi gépen történik, a hálózatot csak az adatátvitelre használjuk fel, illetve a válaszadásra. A megvalósításhoz szintén ágens alapú technológiát

alkalmaznak, mint ahogy azt tesszük mi is munkánkban.

Merdan és szerzőtársai [12] az intelligens ágens alapú technológiát tartják igéretes eszköznek, hogy javítsák olyan munkafolyamatok ütemezését, amelyeknek összetett és gyorsan változó környezetben kell működniük. Vizsgálják az előre nem látható események következményeit, pl. egy gép meghibásodását, amely általában a kezdeti gyártási ütemtervtől való eltérést vonja maga után következményként. A multi-agens megoldást javasolják használni a probléma kezelésére, mivel növelhető a rendszer rugalmassága és megbízhatósága. Munkájukban egy módszert javasolnak a dinamikus rendszer ütemezésre, amellyel a váratlan eseményekre is képesek reagálni. A mi célunk is egy multi-agens rendszer létrehozása volt, amelynek segítségével a munkafolyamathoz kapcsolódó változásokat érzékelve, az erőforrások új kiosztására vonatkozóan egy döntési javaslat születhet a munkafolyamat módosítását illetően.

Ahhoz, hogy ezt a döntési javaslatot a legpontosabban tudjuk megadni, további módszert alkalmaztunk. Mivel a munkafolyamatok aktuális lefutása alatt folyamatosan tárolásra kerülnek a legfontosabb információk egy-egy naplófájlban (logban), ezért ezek segítségével a pillanatnyi állapotnak megfelelő elemzést elvégezhetjük és váratlan események bekövetkezése esetén azonnal reagálhat a multi-agens rendszer egy új erőforrás ütemezési javaslattal. A log elemzés egy viszonylag új kutatási terület, amelynek alkalmazásával rejtett információk feltárására nyílhat lehetőségünk a folyamatok lefutásával kapcsolatban. Van der Aalst és társai úttörő szerepet játszanak a folyamatbányászat területén, lásd pl. [2], [1], [15] munkákat.

A megvalósított munkánkban a felhő alapú technológiát, az ágens alapú megközelítést, a folyamatbányászatot és egy heurisztikus ütemezési módszert integráltunk, amely által egy új megoldást hoztunk létre a munkafolyamatok erőforrás ütemezésének megoldására véletlenszerűen változó végrehajtási környezetben.

2. A komplex feladat bemutatása

Az általam megvalósított feladat egy komplex rendszer részét képezi. Ebben a fejezetben bemutatásra kerül a felhő alapú rendszer, amelynek egyik fő komponense az új megközelítés. Az új szemléletet szeretném bemutatni ebben a dolgozatban.

A komplex feladat fő célja egy olyan munkafolyamatokat kezelő rendszer kidolgozása és megvalósítása, amelyben egy web alapú tervezőfelület segítségével, a BPMN (Business Process Model and Notation)[21] szabványos jelölésrendszerét használva, különböző munkafolyamatokat tudunk létrehozni. Az egyes munkafolyamatokat egy munkafolyamat motor (**Workflow Engine, WFE**) fogja végrehajtani és szimulálni. A munkafolyamatok definiálásakor az egyes feladatok elvégzéséhez nem explicit módon kerülnek hozzárendelésre az erőforrások, hanem a feladat elvégzéséhez szükséges jogosultságok és kvalitások rögzítésével. A hozzárendelés a szimuláció során az adott állapottól függően kerül végrehajtásra.

A folyamat szimulálása során a **Workflow Engine** minden állapothoz előállítja a hozzárendelt erőforrások által elvégzendő feladatot. Ezek megtekintésére egy webes felület (**Task List Manager, TLM**) szolgál, amely képes lebonyolítani a folyamat során szükséges műveleteket. Minden művelet egy dinamikusan generált űrlapon keresztül végezhető el.

A felhasználói döntések támogatásához két különálló modul kerül kifejlesztésre: ágens szimulációs modul (**Agent Simulation Module, ASM**) és a logelemző modul (**Log Analysis Module, LAM**). Az ágens modul kerül bemutatásra ebben a dolgozatban, amely egy adott állapotból kiindulva, a munkafolyamat hátralevő részét szimulálja. Ezek alapján pedig képes ajánlásokat tenni ott, ahol az erőforrások hozzárendeléséről kell dönteni. Így megadja, hogy a szimuláció során melyik választás eredményezi a legjobb futást. A szimulációhoz nyújt segítséget a logelemző modul, amely egy erőforrás paramétereit határozza meg, úgy mint megbízhatóság és hatékonyság.

Az **1. ábrán** látható a rendszer egy sematikus terve, amely a különböző modulokat és azok kapcsolatát szemlélteti. A **Designer** egy webes tervezőfelület, ahol az erőforrások kezelése és a munkafolyamatok építése történik. Ez a modul csak a munkafolyamat adatbázissal van kapcsolatban. Az itt létrehozott munkafolyamatok betölthetők a **Workflow Engine**-be, amely elvégzi a munkafolyamat végre-

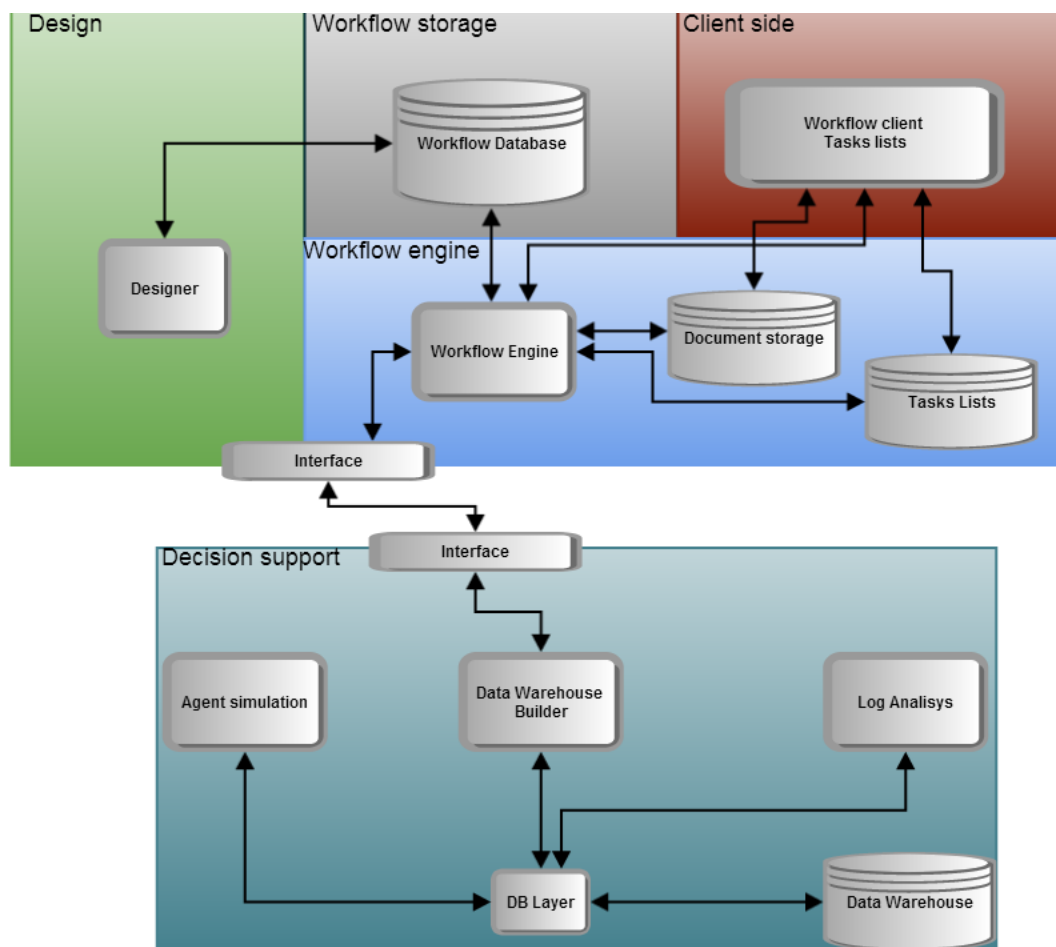
2. A KOMPLEX FELADAT BEMUTATÁSA

hajtását. A szimuláció során egy dokumentumtárat (**Documents Storage**) és egy feladatlistákat tartalmazó adatbázist (**Tasks Lists**) kell kezelni. A munkafolyamatok végrehajtása során az egyes feladatoknál a megfelelő dokumentumba bejegyzés kerül, amely lehet egy időbélyeg digitális aláírással, vagy a feladat eredményének tárolása. A felhasználóknak egy webes kliens áll a rendelkezésükre, amelyeken különböző szempontok szerint listázhatják az elvégzendő feladatokat, valamint egy naptárban jeleníthetik meg a beosztást. A dokumentumok szerkesztése helyben elvégezhető. A kliens minden eseményről visszajelzést ad a munkafolyamat motorának, amely így dönthet arról, hogy a feladat befejezett-e, illetve, hogy a következő feladat végrehajtásához megvannak-e a szükséges feltételek.

A rendszer másik fontos modulja a döntéstámogató rendszer, amely egy adott folyamatot képes leszimulálni, és az egyes döntési szituációkban különböző alternatívákat felajánlani, amelyek a folyamat határidőn belüli befejezését biztosítják. A rendszer rendelkezik saját adatbázissal, amelyben a különböző erőforrásokra és folyamatmodellekre vonatkozó definíciók tárolódnak. A döntéstámogató rendszer egy olyan interfészen keresztül kapcsolódik a másik rendszerhez, amely képes szervertként és kliensként is működni. Ez a megoldás a két rendszer adatbázisának a szinkronizációját biztosítja. Az adatbázis felépítését egy **Data Warehouse Builder** modul biztosítja az interfészen átvett definíciók alapján.

A felhasználói eseményeket a rendszer adatbázisban tárolja, amelyekből a logelemző modul (**Log Analysis**) megbízhatósági profilokat készít az egyes erőforrásokra. Az ágensek allokációja során a rendszer figyelembe veszi, hogy az adott erőforrástól mennyi idő alatt várható a feladat teljesítése.

2. A KOMPLEX FELADAT BEMUTATÁSA



1. ábra. A teljes rendszer sematikus rajza

Egy erőforrásra vonatkozóan a rendszer a következő paramétereket vizsgálja meg:

- Az erőforrás a feladat végrehajtásához szükséges kvalitásokkal és jogosultságokkal rendelkezik-e.
- Az adott erőforrás rendelkezésre áll-e a kívánt időpontban.
- A log alapján készített megbízhatósági profil, ami megadja, hogy a kérdéses erőforrás milyen hatékonysággal fogja elvégezni a feladatot.

A döntéstámogató rendszer által visszaadott eredmények többféleképpen is megjelenhetnek a felhasználói felületen. Néhány lehetséges megvalósítás:

2. A KOMPLEX FELADAT BEMUTATÁSA

- Egy döntési helyzetben a rendszer erőforrásokat ajánl az adott feladat megoldására. Az egyes ajánlatokat képes megindokolni. A dinamikus űrlapon pl. egy rádiógomb segítségével választható ki, hogy melyik erőforrás végezze a kijelölt feladatot. Természetesen az ágens modul által felkínált lehetőség opcionális, így az űrlap lehetőséget biztosít saját döntés meghozatalára is.
- Az ágens modul jelzi, ha egy munkafolyamat az adott állapotból nem végezhető el.
- Amennyiben egy munkafolyamat az adott állapotból nem valósítható meg, akkor a rendszer automatikusan alternatívákat keres a cél elérése érdekében.

Döntéstámogató alrendszer

A döntéstámogató rendszer - melynek részét képezi az ágens szimulációs modul - feladata a külső rendszertől kapott folyamatmodellek és erőforrások saját adatbázisban való tárolása és feldolgozása szimuláció útján. Az eredmény minden esetben egy olyan ajánlás, amely a munkafolyamat egyes feladataihoz a lehető legjobb erőforráhozrendelést teszi lehetővé, biztosítva ezáltal a végrehajtás határidőn belüli teljesítését.

Az **1. ábra** alsó részén látható a döntéstámogató rendszer sematikus tervrajza, amely a különböző komponenseket és az azok közötti kapcsolatot szemlélteti. A teljes rendszer egy interfészen keresztül kapcsolódik az üzleti folyamatokat kezelő rendszerrel (továbbiakban **WMS** - Workflow Management System). Az interfész működését tekintve kliens és szerver is egyben, amely lehetővé teszi a dinamikus adatszinkronizációt az adatbázisok között. Ez azt jelenti, hogy az interfész modul képes fogadni olyan beérkező kapcsolatot, amely a helyi adatbázis frissítésére vonatkozik, valamint képes kliensként saját maga is lekérdezni a külső adatbázisok állapotát. Az interfészen keresztül fogadott folyamatmodellek definíciói és az erőforrásdefiníciók egy *JSON* struktúra formájában kerülnek átadásra, amelyet a **Data Warehouse Builder** modul értelmez. A kapott adatok alapján pedig felépíti az adatbázist. A felépített adatbázisnak minden információt tartalmaznia kell az erőforrásokról, a folyamatokról és a működés során keletkezett logokról. A logelemző modul (**Log Analysis**) feladata az adatbázisban tárolt információk alapján, a **WMS** által feltett kérdésekre való válaszadás. Ilyen kérdés lehet például, hogy adott fo-

2. A KOMPLEX FELADAT BEMUTATÁSA

lyamat, adott lépésében, a jelenlegi foglaltságok mellett mi az optimális erőforrás hozzárendelés. Az általam fejlesztett ágens alapú szimulációs modul (**Agent Simulation**) felelős a tárolt folyamatmodellek és erőforrásdefiníciók értelmezéséért és szimulációjáért. Konkrétan, a modul egy folyamatot szimulál, és a teljes folyamatra adja vissza, hogy a log-elemzés és az adatbázis alapján melyik erőforrás hozzárendelés esetén lesz optimális a folyamat futása.

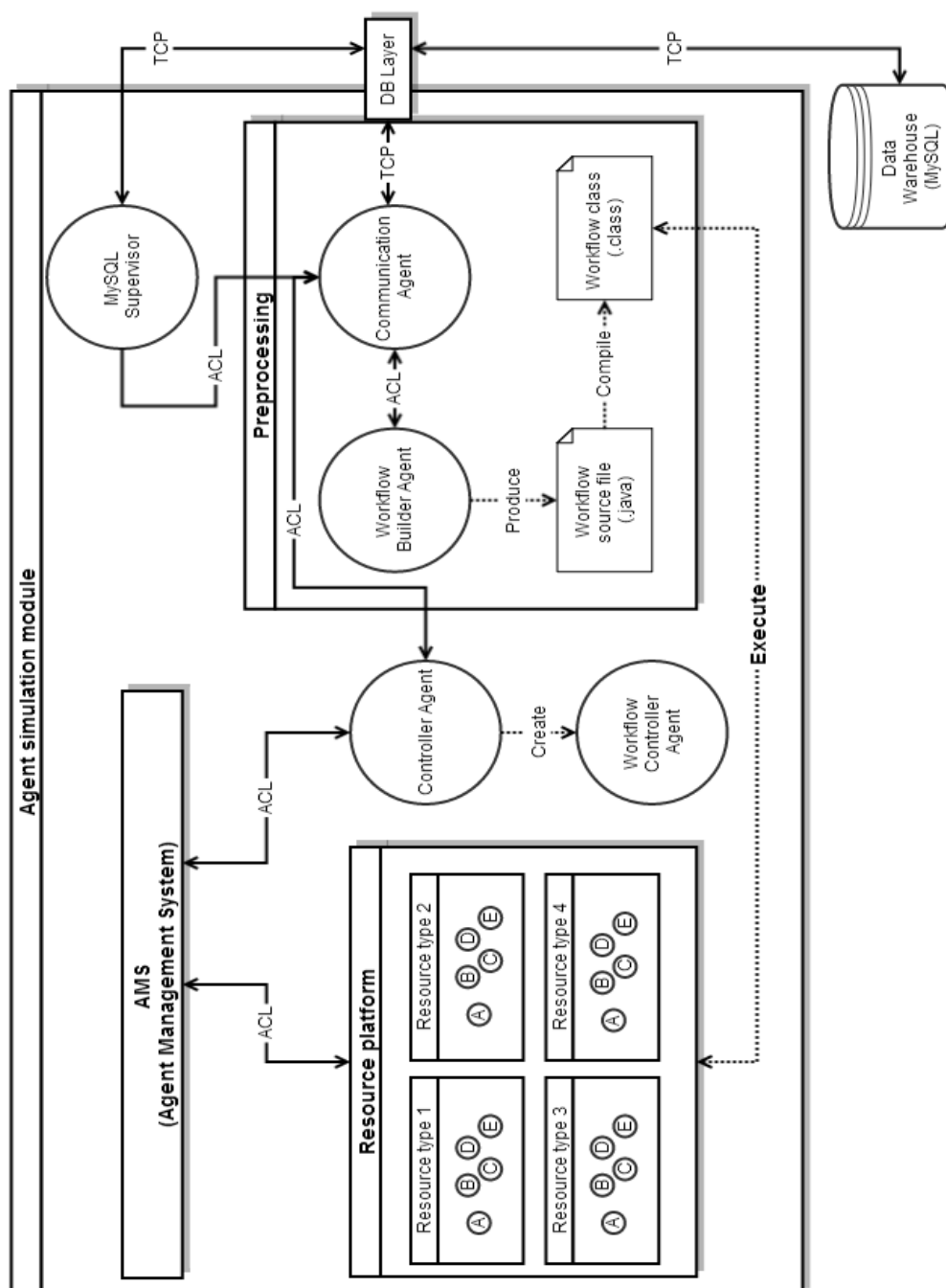
3. Döntéstámogató alrendszer

3.1. Ágens alapú szimulációs modul

Az ágens alapú szimulációs modul egy multi-ágens rendszerként lett felépítve, amely felhasználja a hálózati kapcsolatokat, a szimulációs megoldást, folyamatbányászati technikát, valamint egy általunk kifejlesztett erőforrás ütemezési megoldást.

A létrehozott multi-ágens rendszer WADE [22] környezetben lett kifejlesztve, amely egy munkafolyamat- és ágensfejlesztő rendszer. A WADE által biztosított ágens platform segítségével tudtuk az ágenseket magas szinten, hatékonyan kezelni. Az implementációt megelőzően modelleztük a rendszer működését, amelyhez az elkészült modellek a következőkben kerülnek bemutatásra. A **2. ábrán** a szimulációs modul felépítése látható. A modul két fő részből épül fel; az előfeldolgozó egység és maga a szimulációs környezet. Az előfeldolgozó egység feladata - a szimulációt előkészítendő - a szükséges adatok áttöltése az adatbázisból, majd ezekből az információkból a megfelelő forrásfájlok előállítása és lefordítása. Miután a munkafolyamatokat leíró állományok rendelkezésre állnak, a második szakaszban egy vezérlő ágens (*Controller Agent*) automatikusan elindítja az ágens platformot. Ebben a vonatkozásban platform alatt az ott futó ágensek és az ezeket tartalmazó tárolók összességét értjük. A platform indításakor csak az alapvető, rendszerszintű tárolók és a hozzájuk tartozó felügyelő ágensek kerülnek elindításra. A sikeres indítást követően a *Controller Agent* az adatbázisból átvett erőforrásdefiníciók alapján megkezd a platform felépítését; létrehozza a megfelelő tárolókat, amelyek mindegyike egy erőforrástípusnak (azaz azonos típusú erőforrások halmazának) felel meg, majd ezután az erőforrásleírás alapján allokalja a megfelelő számú ágens a tárolókban, amelyek a konkrét erőforrásokat reprezentálják.

A rendszer lehetőséget nyújt arra is, hogy esetlegesen a szimuláció közben is lehessen módosítani a platform szerkezetét (pl. új ágensek indítása vagy már meglévők migrálása tárolók között stb.). Ez a funkció az újraütemezéskor (erőforrások helyettesítése) lehet hasznos.

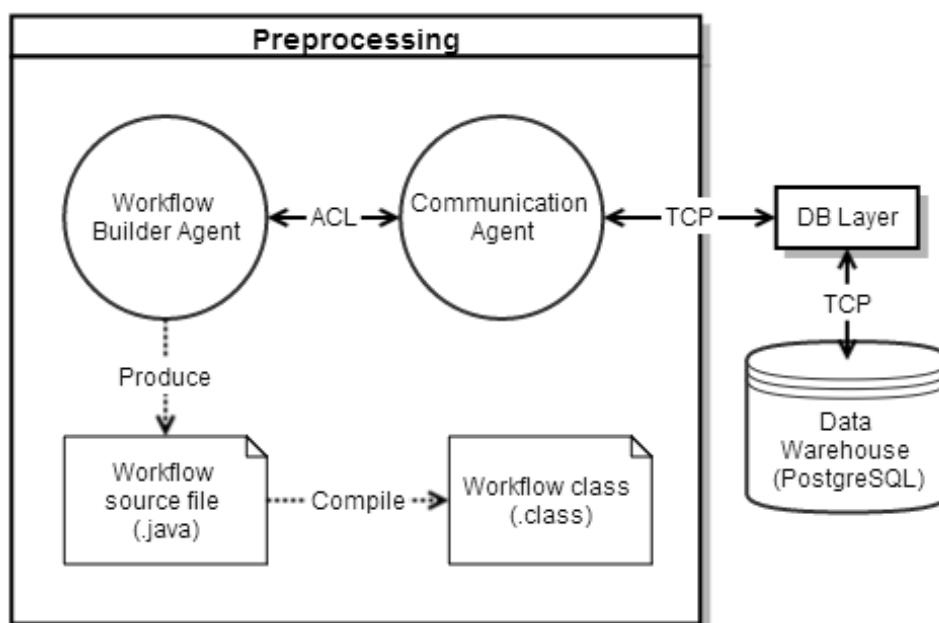


2. ábra. Általános modell

A következő pontokban részletesen bemutatom a modell egyes komponenseit.

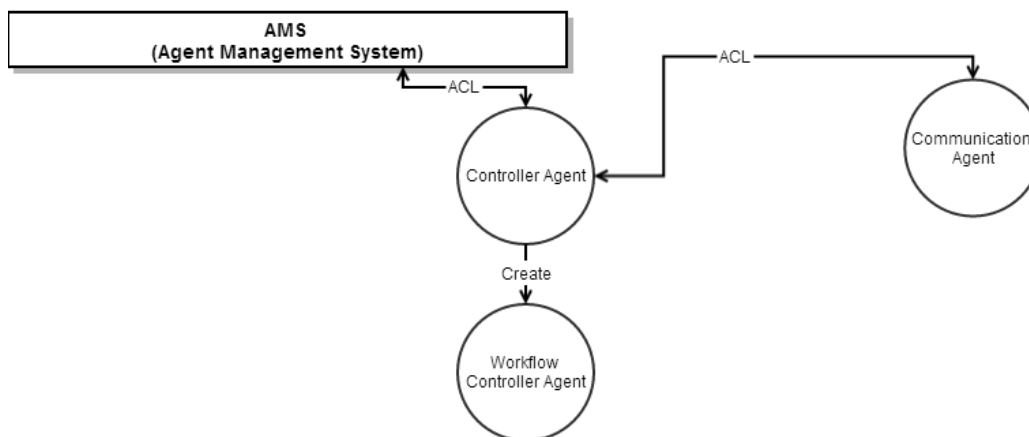
3.1.1. Előfeldolgozó (Preprocessing)

Az előfeldolgozó egység, amely a **3. ábrán** látható, két ágenszt foglal magába. Az adatbázissal való kommunikációért a *Communication Agent* lesz a felelős, melynek feladata a folyamatmodelleket és az erőforrásokat leíró definíciók lekérdezése. Az adatok lekérdezése után a *Workflow Builder Agent* a folyamatmodellekre vonatkozó információkat hasznosítja. A definíciókból automatikusan legenerálja a munkafolyamatokat leíró *.java* forrásfájlokat. A forrásfájlok ezután lefordításra kerülnek, hiszen csak így lesznek futtathatóak az ágensek számára. A lefordított állományok egy előre kijelölt könyvtárban tárolódnak, ahonnan majd a végrehajtás során betölthetők. Összegezve, az előfeldolgozási folyamat feladata, hogy a szimulációs környezet számára megfelelően előkészítse a végrehajtandó munkafolyamatot.



3. ábra. Előfeldolgozás modellje

3.1.2. Vezérlő ágens (Controller Agent)

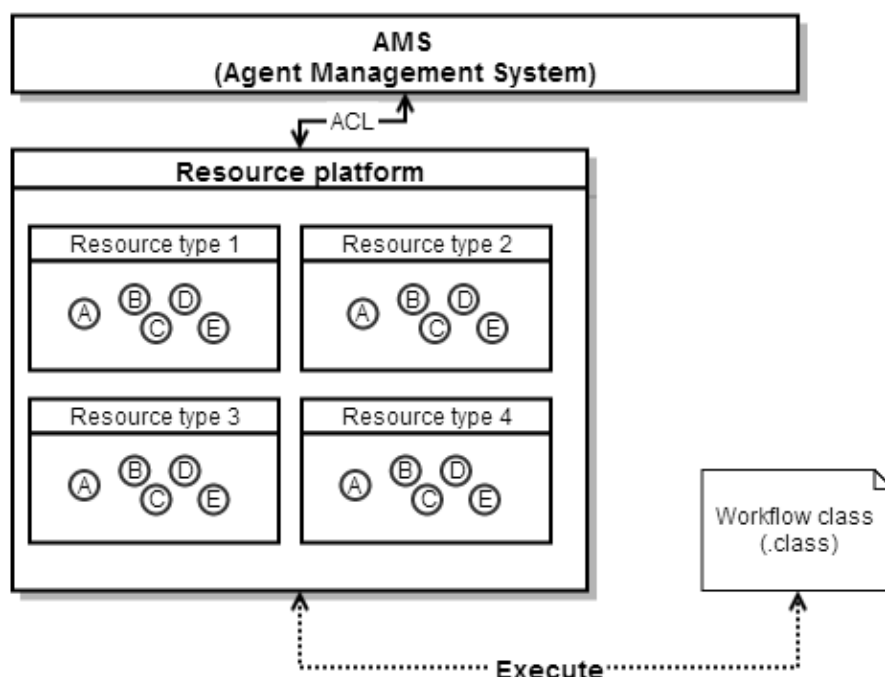


4. ábra. Vezérlő ágens (Controller Agent)

A vezérlő ágens feladata az adatbázisból nyert erőforrásdefiníciók feldolgozása, amelyekhez a *Communication Agent* közreműködésével jut. A feldolgozási folyamat során az ágens az adatbázisból letöltött erőforrásleírás alapján felépíti a platformot. A **4. ábrán** látható, hogy a vezérlő ágens kapcsolatban van egy AMS [24] modullal. Az AMS egy platformfelügyelő alrendszer, amely a keretrendszer része, és felelős a platformon lezajló események felügyeletéért, valamint megfelelő kérelem után képes futási időben módosítani a platform szerkezetét. A módosítás lehet egy új konténer indítása, új ágens létrehozása, már meglévő ágens migrálása másik konténerbe stb. A *Controller Agent* az AMS-nek küldött kérések segítségével alakítja ki a platform szerkezetét a szimulációhoz. Továbbá, mivel a *Controller Agent* feladata a szimuláció indítása, így minden szimulációhoz létrehoz egy azt kezelő ágens is (*Workflow Controller Agent*).

3.1.3. A platform

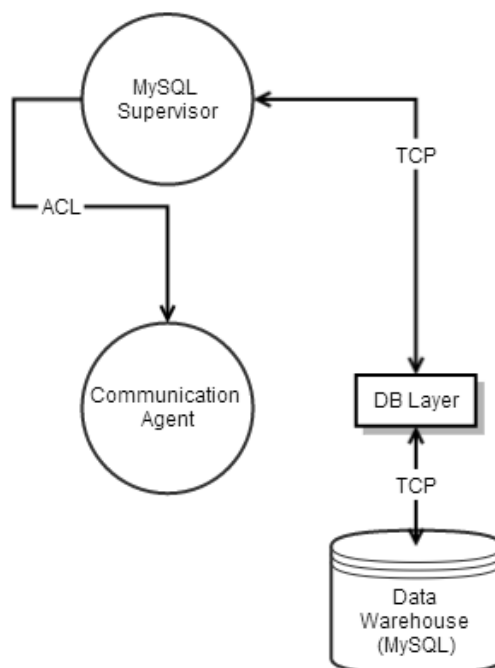
Minden egyes tároló egy erőforrástípusnak felel meg, amelyekben az ágensek jelentik a konkrét erőforrásokat. Az erőforrásokat reprezentáló ágenseknek a feladata az előfeldolgozás során létrehozott munkafolyamat lejátszása.



5. ábra. A platform általános felépítése

3.1.4. Adatbázis felügyelet

A **6. ábrán** látható a *MySQL Supervisor* ágens, amelynek feladata a MySQL adatbázis elérhetőségének folyamatos felügyelete. Az ágens két másodpercenkénti periódusidővel teszteli a MySQL adatbázissal való hálózati kapcsolat állapotát. Ha az adatbázis bármilyen ok miatt nem elérhető (hálózati hiba, az adatbázis nem fut stb.), akkor automatikusan jelzi ezt a felhasználó irányába a grafikus felületen (lásd 3.9 alfejezet), illetve az érdekelt ágensok részére felfüggeszti a hálózati kapcsolatot. Mivel az adatbázis elérhetősége a rendszer működésének egyik alapja, így annak kiesése esetén a rendszer készenléti állapotba kerül mindaddig, amíg a kapcsolat helyre nem áll.



6. ábra. Adatbázis elérhetőségének felügyelete

3.2. Kommunikáció, protokoll, automatizált mechanizmusok

3.2.1. Alkalmazott kommunikációs protokollok

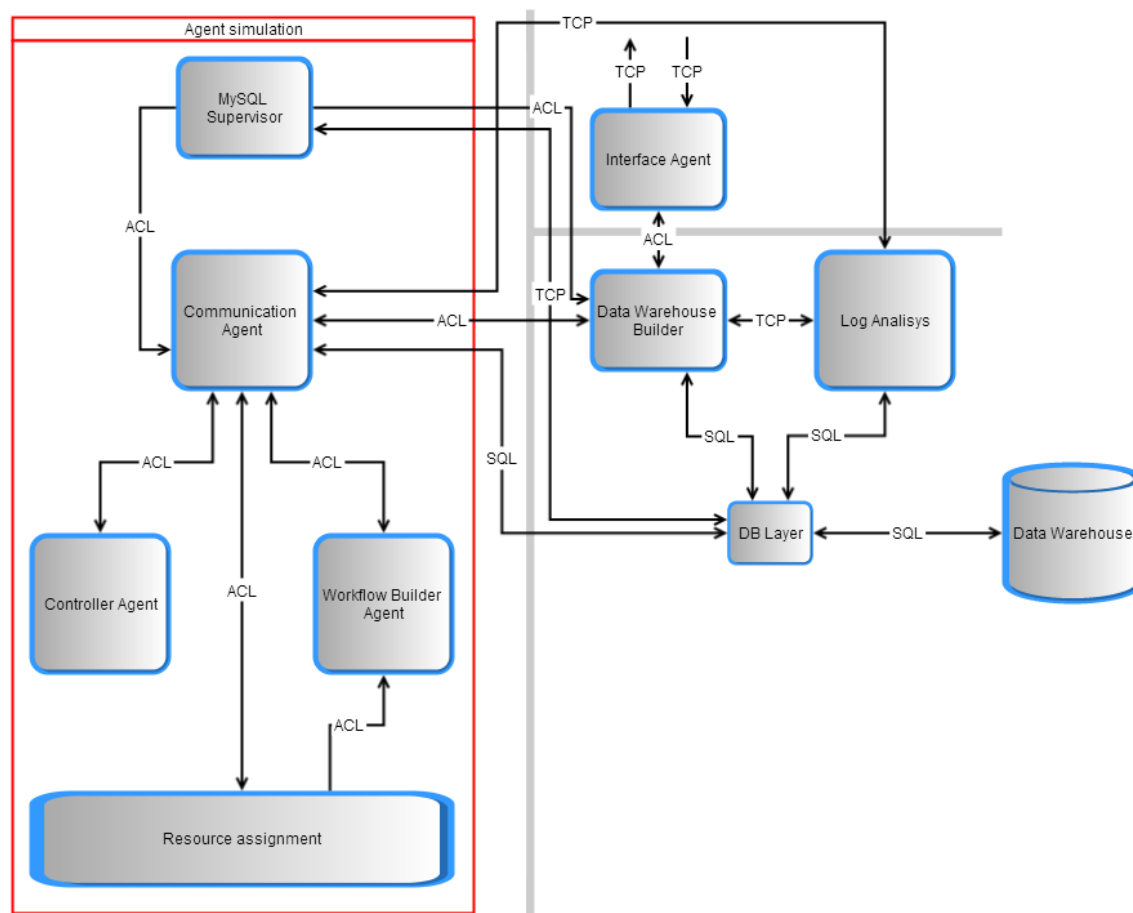
A következőkben bemutatásra kerülnek azok a protokollok, amelyeket a rendszer alkalmaz a kommunikáció során. Az alkalmazott protokollok a *TCP*, az *ACL* és az adatbázishoz tartozó lekérdező nyelv, az *SQL*.

3.2.2. Kommunikációs modell

A **7. ábrán** láthatók a rendszerben működő ágens modulok közötti kommunikációs csatornák. A döntéstámogató rendszer önálló működésének alapja, hogy az egyes modulok megfelelően tudjanak egymással kommunikálni.

Az első szekcióba tartozik az interfész ágens (*Interface Agent*), amelynek feladata a külső rendszerrel való kommunikáció. A modul kliensként és szerverként is működik; kliensként kapcsolódik a külső rendszerhez, szerverként pedig a külső rendszerből érkező kapcsolatokat kezeli. A kimenő és beérkező kapcsolatok a *TCP* hálózati protokoll szerint működnek. Az *Interface Agent* rendszeroldalon egy *ACL*

3. DÖNTÉSTÁMOGATÓ ALRENDSZER



7. ábra. Kommunikációs modell

protokoll szerint működő csatornát tart fenn, amelyen keresztül átadja a rendszernek a folyamatmodell és az erőforrások leírását. A második szekcióban két modul kapott helyet. A *Data Warehouse Builder* feladata az interfész ágenstől érkező munkafolyamat definícióinak az átvétele, majd az adatbázisba való áttöltése, továbbá az egyes munkafolyamatok futási eredményei is ezen a modulon keresztül kerülnek vissza az interfész ágenshez, amely a külső rendszernek továbbítja azokat. A log analizáló (*Log Analysis*) modul feladata a historikus adatok feldolgozása az egyes folyamatmodellek futásáról, amely információkból különböző tulajdonságok állapíthatók meg, mint pl. az egyes erőforrások hatékonysága és megbízhatósága. A *Data Warehouse Builder* és a *Log Analysis* modul *TCP* kapcsolaton keresztül kommunikálnak, amelynek fő célja, hogy értesítsék egymást az adatbázis állapotáról.

A harmadik szekcióban azok az ágensek szerepelnek, amelyek már a szimulációs modul részei. A *Communication Agent* felelős azért, hogy az adatbázisban tárolt folyamatmodellek áttöltésre kerüljenek a szimulációs modulba. Az ágens csak abban az esetben kezdi meg az adatok letöltését, amikor azok készen állnak az adatbázisban. Erről a *Data Warehouse Builder* ágens tájékoztatja egy üzenet formájában, miután betöltötte az adatokat az adatbázisba. A *Communication Agent* ezután a letöltött adatokat továbbadja a *Controller Agent*-nek a *Workflow Builder Agent*-nek és a *Scheduler Agent*-nek. A *Controller Agent* az erőforrásdefiníciók alapján felépíti a platformot a szimulációhoz, amely a tárolók létrehozását és az erőforrások fizikai lefoglalását jelenti a rendszerben. A *Workflow Builder Agent* feladata a folyamatdefinícióknak megfelelő munkafolyamat (*workflow*) felépítése.

A munkafolyamatok felépítése egyben magában foglalja az erőforráskiosztás mechanizmusát is, ugyanis a munkafolyamatok felépítése során már kötelezően meg kell adni a kiosztás eredményét. Az erőforrások kiosztását a *Scheduler Agent* végzi, és a fentiek értelmében az ütemezés megelőzi a munkafolyamatok összeállítását. Az ütemezés befejeztével az ütemező ágens egy üzenet formájában adja át a *Workflow Builder Agent*-nek az eredményeket. Utóbbi a kiosztásnak megfelelően állítja össze a munkafolyamatokat. Miután a munkafolyamatok legenerálásra kerültek, a *Controller Agent* automatikusan indítja a szimulációt, amely már az ütemezett erőforrások segítségével fut le.

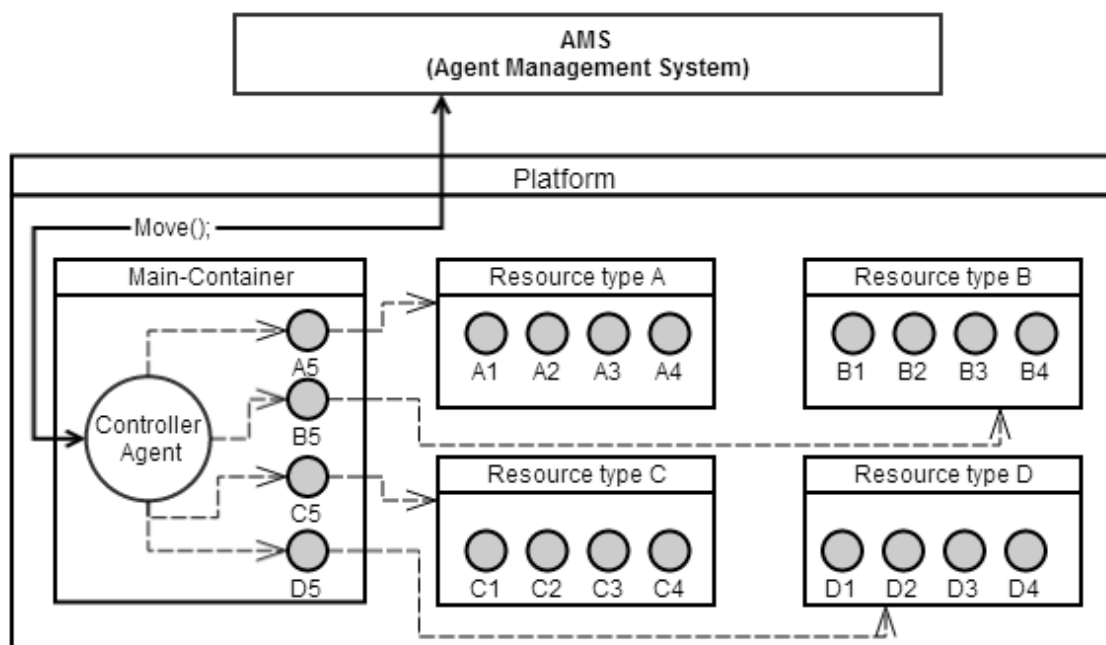
Egy munkafolyamat lejátszását követően a futási eredmények automatikusan visszatérnek a *Workflow Controller Agent*-hez, amely továbbítja ezeket a *Controller Agent*-nek. A *Controller Agent* pedig továbbadja az információkat a *Communication Agent*-nek, amely visszatölti az eredményeket az adatbázisba. Innen a *Data Warehouse Builder Agent* tölti le az adatokat, majd az *Interface Agent* közreműködésével visszajuttatja azokat a külső rendszerhez.

3.3. Erőforrások kezelése

A rendszerben működő erőforrások kezeléséért közvetett módon a *Controller Agent* a felelős, amely az adatbázisból nyert adatok alapján biztosítja az erőforrások allokációját. A rendszerben az erőforrástípusokat a tárolók reprezentálják, amelyekben a megfelelő számú ágensek, mint erőforrások kerülnek lefoglalásra. Minden lefoglalt ágens egy micro-workflow motort valósít meg, amelynek köszön-

3. DÖNTÉSTÁMOGATÓ ALRENDSZER

hetően válnak képessé a munkafolyamatok kezelésére. A következőkben részletesen ismertetem az erőforrások kezelésének technikáját.



8. ábra. Erőforrások allokációjának a modellje

A **8. ábrán** látható, hogy a rendszer milyen formában végzi az erőforrások foglalását. Első lépésben a *Controller Agent* létrehozza az adott erőforrástípus-hoz tartozó tárolót, amelynek elnevezése természetesen megegyezik az erőforrás-típus nevével. A tároló létrehozását követően az erőforrásdefinícióban megadott számú konkrét erőforrást foglal le. A lefoglalt erőforrások alapértelmezésként a *Main-Container*-ben jönnek létre. Emiatt az allokációt követően egy migrációra van szükség, amely eljárásban az erőforrások a megfelelő tárolókba kerülnek.

Látható, hogy a *Controller Agent* egy *AMS* alrendszerrel van kapcsolatban. Az *AMS (Agent Management System)* egy ágenskezelő és felügyelő rendszer. Az *AMS* végzi az ágensek migrációját az egyes tárolók, konkrétan a *Main-Container* és a céltároló között. Természetesen az *AMS* kérelem alapú szolgáltatásokat biztosít, ami azt jelenti, hogy egy erőforrás áthelyezését a *Controller Agent*-nek kell kérelmeznie. A migrációs folyamat elfogadásáról vagy elutasításáról az *AMS* minden esetben visszajelez.

3.4. Folyamatmodellek feldolgozása

3.4.1. Munkafolyamat megadása absztrakt formában

A szimulációs modul másik legfontosabb bemenete az erőforrásokon túl a munkafolyamatok, vagy ismertebben a *workflow*. A munkafolyamatok a való életben működő folyamatokat, többek között üzleti vagy gyártási folyamatokat modelleznek.

Minden munkafolyamat tevékenységekből áll, illetve a tevékenységek közötti függőségekből. A tevékenységek jelentik egy folyamat építőelemeit, amelyben egy vagy több művelet foglal helyet. A műveletek határozzák meg egy tevékenység szintjén az elvégzendő feladatokat. Az egyes tevékenységeket élek kötik össze egymással, amelyek egyértelműen meghatározzák a végrehajtás irányát. A munkafolyamatok a legegyszerűbb szekvenciális felépítéstől a párhuzamos tevékenységekig gyakorlatilag bármilyen szerkezetet mutathatnak.

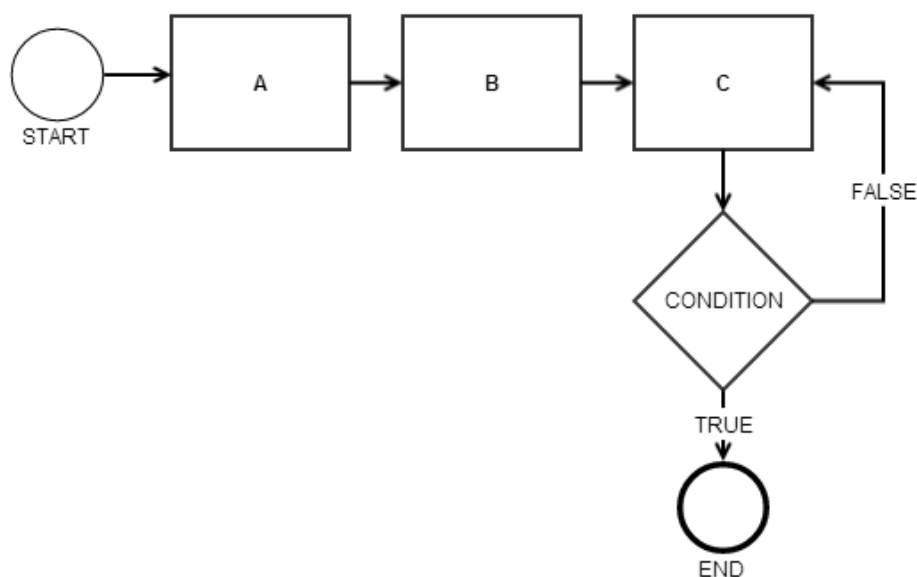
A fentiek alapján minden munkafolyamat egy irányított gráfként írható le, így az irányított gráf definíciójának [27] felhasználásával azt mondhatjuk, hogy a munkafolyamaton egy olyan

$$G = (V, E, \varphi)$$

hármast értünk, ahol a V halmaz elemeit a gráf pontjainak (munkafolyamat tevékenységeinek), az E halmaz elemeit a gráf éleinek (a munkafolyamat tevékenységei közötti kapcsolatoknak) nevezzük. A φ pedig tetszőleges $E \rightarrow V \times V$ leképezés.

3.4.2. Munkafolyamatok a döntéstámogató alrendszerben

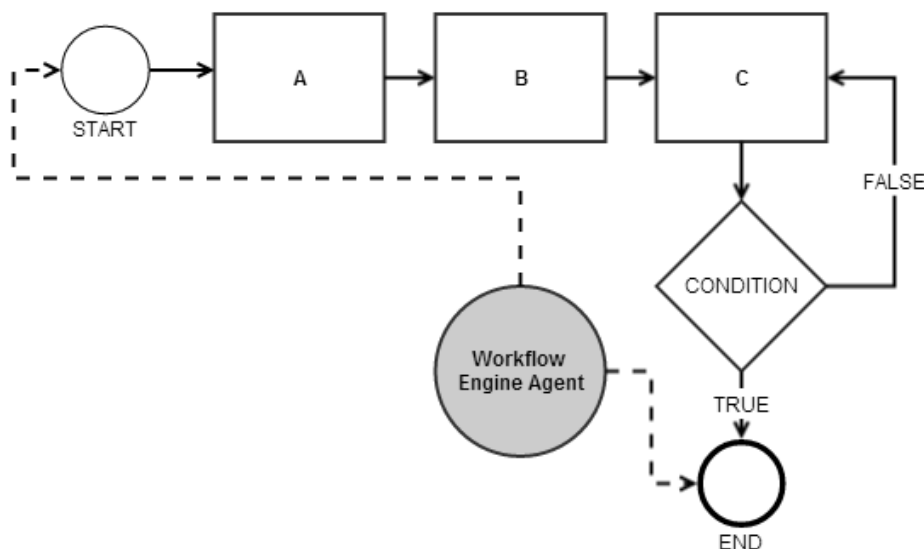
A döntéstámogató alrendszer implementációja a *WADE* keretrendszerre épül. Ennek köszönhetően a munkafolyamatok kezelése is a keretrendszer szabályainak megfelelően történik. Ahogy az előző pontban láthattuk, a munkafolyamatok egy egyszerű matematika szabállyal leírhatóak. A következőkben ismertetésre kerül, hogy a tényleges rendszerben miként működnek a munkafolyamatok, illetve kitérünk azokra a problémákra, amelyeket a keretrendszer határainak kiterjesztésével kellett megoldani.



9. ábra. Egyszerű munkafolyamat

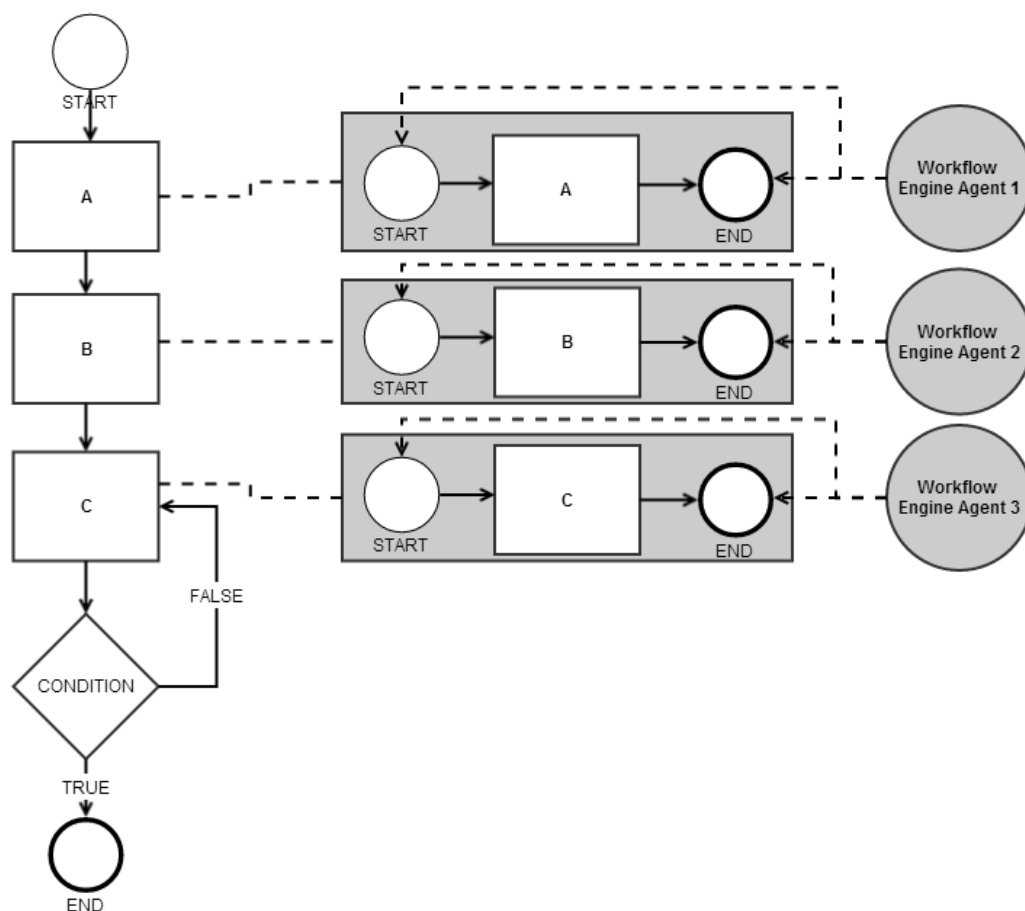
Tekintsük a **9. ábrát**, ahol egy egyszerű munkafolyamat látható. Összesen három (*A*, *B*, *C*) tevékenységből épül fel, ahol az *A* tevékenység a kezdőállapot, a *C* pedig a feltételtől függően lehet köztes állapot vagy végállapot.

A döntéstámogató alrendszer a folyamatmodelleket természetesen nem ilyen magasszintű formában tárolja, hanem *.java* osztályok segítségével. A keretrendszer alapértelmezetten támogatást nyújt ehhez. A beépített funkciókat (osztályok, metódusok, speciális változók stb.) alkalmazva könnyedén tudja a rendszer kezelni a folyamatokat. Mivel a megvalósított döntéstámogató rendszerben működő szimulációs modul legfontosabb feladata az erőforrások optimális ütemezése, valamint a folyamatok erőforrásokkal történő végrehajtása, így természetesen szükséges a munkafolyamatokhoz erőforrásokat rendelni.



10. ábra. Munkafolyamat végrehajtása egy ágenssel

A **10. ábrán** látható az előbbieken bemutatott (9. ábra) munkafolyamat, immáron egy *workflow* motorként működő ágenssel. A szaggatott vonallal megrajzolt nyilak azt mutatják, hogy ez az ágens a teljes munkafolyamat végrehajtásáért felelős, a belépési ponttól egészen a kilépési pontig. Ez a végrehajtási stratégia sajnos nem megfelelő, hiszen teljesen szemben áll az erőforrások ütemezésével. Ebben az esetben a teljes munkafolyamat végrehajtása egy ágensre van bízva. Az ütemezhetőség szempontjából nem a teljes munkafolyamatot kell tekinteni, mint egészet, hanem a tevékenységek szintjén kell keresni a megoldást. A célunk az volt, hogy egy munkafolyamaton belül minden egyes tevékenységhez külön erőforrásokat rendeljünk, így az alapértelmezett "*egy főfolyamat, egy ágens*" elv nem volt használható. A megoldás érdekében szükségessé vált a munkafolyamatok kezelésének a kiterjesztése. Az átdolgozott modell a **11. ábrán** látható.

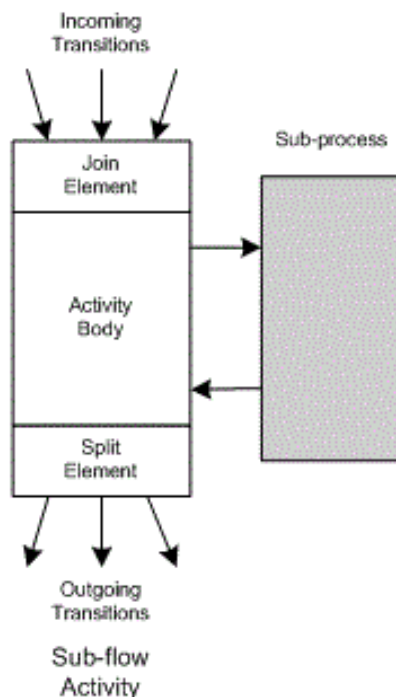


11. ábra. Munkafolyamat végrehajtása több ágenssel

A munkafolyamatok végrehajtását olyan módon kellett megvalósítani, ahol nem a teljes munkafolyamathoz rendel hozzá a rendszer egy erőforrást, hanem a munkafolyamaton belül minden tevékenységhez külön erőforrás tartozik. Ennek érdekében szükségszerűvé vált bevonni a kretrendszer által is támogatott alfolyamatokat.

Az alfolyamatok olyan speciális részei a főfolyamatnak, amelyek a főfolyamaton belül az egyes tevékenységekbe épülnek be. Ahogy a **11. ábrán** látható, a tényleges főfolyamat minden egyes tevékenysége (A, B és C) egy alfolyamatot rejt. Az alfolyamatok is teljesértékű munkafolyamatnak felelnek meg és ebben a megvalósításban csak azt a műveletet vagy műveleteket tartalmazzák, amelyet a főfolyamból hozzájuk tartozó tevékenység is tartalmazna. Mivel minden egyes alfolyamat különálló munkafolyamatot jelent, így mindegyikhez külön erőforrást kell hozzárendelni a végrehajtáshoz. Ezzel a stratégiával sikerült is megoldani az ütemezhetőség prob-

lémáját. A következőkben a munkafolyamatokhoz kapcsolódóan ismertetem azok működését a megoldást jelentő modell (11. ábra) alapján.



12. ábra. Alfolyamattá kiterjesztett tevékenység [28]

Ahogy fentebb is szó volt róla, az alfolyamatok a főfolyamat tevékenységeibe épülnek be. Minden alfolyamat alapértelmezetten szinkron módban működik, amely azt jelenti, hogy a főfolyamat blokkolódik minden tevékenységnél, amíg az alfolyamat fut. Amint az alfolyamat befejeződött, úgy a főfolyamat futása folytatódik. Ennek modellje a **12. ábrán** látható.

3.5. Ütemezés

Az ütemezési eljáráshoz szükséges volt egy olyan algoritmus kidolgozása, amely az egyes munkafolyamatokban előforduló feladatokhoz (*task*) a rendelkezésre álló erőforrások közül a lehető legjobbat rendeli hozzá. Az eljárás két irányelvet követhet: idő- vagy költségalapú ütemezés.

3.5.1. Általános ütemezési modell

Legyenek $Process$, $Task$, $Resource$, $ApplicableResource$, P , O halmazok, ahol

$$\begin{aligned} Process &= \{process_1, process_2, \dots, process_n\}, \\ Task(process_i) &= \{t_{i1}, t_{i2}, \dots, t_{im}\} \quad 1 \leq i \leq n, \\ Resource &= \{r_1, r_2, \dots, r_k\}, \\ ApplicableResource(t_{ij}) &= \{r_{(ij)_1}, r_{(ij)_2}, \dots, r_{(ij)_l}\} \subseteq Resource \\ 1 \leq j \leq m, \quad 1 \leq l \leq k \end{aligned}$$

$$\begin{aligned} P &= \{p_1, p_2, \dots, p_h\}, \\ O(r_q) &= \{o_{q1}, o_{q2}, \dots, o_{qz}\}, \end{aligned}$$

Legyenek rap , t és c függvények, ahol

$$\begin{aligned} rap &: Resource \times Task \times P \rightarrow \{0, 1\} \\ t &: Resource \times Task \times P \times O \rightarrow N \\ c &: Resource \times Task \times P \times O \times N \rightarrow Q \end{aligned}$$

$Process$ a rendszerbe betöltött folyamatmodellek halmaza, $Task(process_i)$ a rendszerbe betöltött i . munkafolyamatban szereplő feladatok halmaza, $Resource$ a rendszerben allokált erőforrások halmaza, amely az összes erőforrást jelenti, munkafolyamattól függetlenül. Az $ApplicableResource(t_{ij})$ az i . munkafolyamat, j . feladatához kvalitás alapján alkalmazható erőforrások halmaza, a P a terméktípusok halmaza, amelyekkel az egyes erőforrások dolgoznak, az $O(r_q)$ pedig a működési módok halmaza, amely a q . erőforrásra jellemző.

Az rap függvény megadja, hogy egy adott munkafolyamaton belül, egy adott feladat, amelyhez valamilyen terméktípus tartozik, melyik erőforrással végezhető el, és melyikkel nem. A t függvény meghatározza, hogy egy adott erőforrás esetén, adott működési módban, adott tevékenységet végrehajtva egy meghatározott terméktípuson az mennyi időbe kerül. A c függvény meghatározza a művelet költségét, amelyet egy adott erőforrás hajt végre egy adott feladatban, adott működési módban, adott terméktípuson, adott végrehajtási idővel.

3.5.2. Időalapú ütemezés

A munkafolyamatok tevékenységei és az erőforrások ütemezése során a hozzárendelési tényező az idő. Az ütemező modul ebben az esetben az egyes tevékenységekhez olyan, kvalitásban megfelelő erőforrást keres, amely a legrövidebb időn belül képes végrehajtani a feladatot, amelynek időigénye természetesen függ a terméktípustól is, amelyen az erőforrás dolgozik.

Legyen egy tetszőleges $t_{ij} \in Task(process_i)$ ($1 \leq i \leq n, 1 \leq j \leq m$) elem, amely a j . tevékenységet reprezentálja az i . munkafolyamatból. Ekkor ezen tevékenység végrehajtására alkalmas erőforrásokat a $ApplicableResource(t_{ij})$ halmaz tartalmazza. Ezen halmaz számítása az rap függvény alapján történik. Legyen $(ApplicableResource(t_{ij}), \leq)$ egy teljesen rendezett halmaz, így ez a halmaz tartalmazza az összes erőforrást - amelyek képesek végrehajtani a $t_{ij} \in Task(process_i)$ feladatot - immár növekvő sorrendben úgy, hogy a rendezés az egyes erőforrásokhoz tartozó t függvényértéke alapján történik. Így $\forall r \in ApplicableResource$ esetén teljesül, hogy $r_{(ij)_1} \leq r_{(ij)_2} \leq \dots \leq r_{(ij)_l}$, és látható, hogy ebben az esetben a $t_{ij} \in Task(process_i)$ feladat végrehajtásához az időalapú ütemezés szerint az $r_{(ij)_1}$ erőforrás a legalkalmasabb.

3.5.3. Költségalapú ütemezés

A munkafolyamatok tevékenységei és az erőforrások ütemezése során a hozzárendelési tényező a költség. Az ütemező modul ebben az esetben az egyes tevékenységekhez olyan, kvalitásban megfelelő erőforrást keres, amely a legkevesebb költséggel képes végrehajtani a feladatot.

Legyen egy tetszőleges $t_{ij} \in Task(process_i)$ ($1 \leq i \leq n, 1 \leq j \leq m$) elem, amely a j . tevékenységet reprezentálja az i . munkafolyamatból. Ekkor ezen tevékenység végrehajtására alkalmas erőforrásokat a $ApplicableResource(t_{ij})$ halmaz tartalmazza. Ezen halmaz számítása az rap függvény alapján történik. Legyen $(ApplicableResource(t_{ij}), \leq)$ egy teljesen rendezett halmaz, így ez a halmaz tartalmazza az összes erőforrást - amelyek képesek végrehajtani a $t_{ij} \in Task(process_i)$ feladatot - immár növekvő sorrendben úgy, hogy a rendezés az egyes erőforrásokhoz tartozó c függvényértéke alapján történik. Így $\forall r \in ApplicableResource$ esetén teljesül, hogy $r_{(ij)_1} \leq r_{(ij)_2} \leq \dots \leq r_{(ij)_l}$, és látható, hogy ebben az esetben a $t_{ij} \in Task(process_i)$ feladat végrehajtásához a költségalapú ütemezés szerint az

$r_{(ij)_1}$ erőforrás a legalkalmasabb.

3.6. Algoritmusok

Algorithm 1 Resource allocation algorithm

```
1: procedure RESOURCEALLOCATION
2:   resourceTypeList
3:   Allocate resources for each resource type:
4:   for  $i:=1$  to resourceTypeList.size do
5:     createContainer(ResourceTypeName)
6:     for  $j:=1$  to resourceTypeList(i).getResourcePiece do
7:       createAgent(ResourceTypeName + "j+1")
8:       moveAgent(ResourceTypeName, ResourceTypeName + "j+1")
9:     end for
10:  end for
11: end procedure
```

A 2. sorban a *resourceTypeList* tartalmazza az összes erőforrástípust a rendszerben. Minden erőforrástípusról rendelkezésre áll, hogy abból hány darabot kell allokálnia a rendszernek. A 4. sorban induló *for* ciklus végighalad az összes erőforrástípuson. Elsőként (5. sor) létrehozza a megfelelő konténert az erőforrástípus nevével. A második lépésben (7. sor) fizikailag létrehozza az erőforrástípushoz tartozó, megfelelő mennyiségű erőforrást. Végül az allokált erőforrásokat a megfelelő konténerbe mozgatja (8. sor).

Algorithm 2 Workflow handling algorithm

```
1: procedure WORKFLOWHANDLING(WORKFLOWS)
2:   processList  $\leftarrow$  Workflows
3:   taskList  $\leftarrow$  null
4:   Handling processes:
5:   for  $i:=1$  to processList.size do
6:     taskList  $\leftarrow$  createTaskList(processList(i))
7:     generateMainWorkflowFile(taskList)
8:     generateSubWorkflowFiles(taskList)
9:   end for
10: end procedure
```

Az eljárás feladata a kapott folyamatmodellekből a rendszer számára futtatható

munkafolyamatok előállítására. A függvény bemenetként megkapja a rendszerbe betöltött összes folyamatmodellt. Egy *for* cikluson felül első lépésben (6. sor) automatikusan legenerálja a rendszer az *i.* munkafolyamathoz tartozó *task*-ok listáját, és a folyamatmodell alapján kiépíti közöttük a kapcsolatokat. A 7. és 8. sorokban pedig az előbbieket alapján legenerálja a szükséges *.java* fájlokat, amelyek a munkafolyamatot reprezentálják feldolgozható formában.

Algorithm 3 Scheduling algorithm for time

```
1: procedure SCHEDULINGFORTIME
2:   processes
3:   resources
4:   applicableResources
5:   for i:=1 to processes.size do
6:     for j:=1 to processes(i).tasks.size do
7:       for k:=1 to resources.size do
8:         if rap(resources(k), processes(i).tasks(j), processes(i).tasks(j).productType) = 1
then
9:           applicableResource(l)  $\leftarrow$  resources(k)
10:          l  $\leftarrow$  l + 1
11:        end if
12:      end for
13:      mT  $\leftarrow$  1
14:      for n:=2 to applicableResource.size do
15:        if t(applicableResource(n)) < applicableResource(mT) then
16:          mT  $\leftarrow$  n
17:          //mT is the index of resource with minimum execution time
18:        end if
19:      end for
20:      //Schedule applicableResource(mT) to tasks(j)
21:      mT  $\leftarrow$  1
22:    end for
23:  end for
24: end procedure
```

Algorithm 4 Scheduling algorithm for cost

```

1: procedure SCHEDULINGFORCOST
2:   processes
3:   resources
4:   applicableResources
5:   for  $i:=1$  to processes.size do
6:     for  $j:=1$  to processes(i).tasks.size do
7:       for  $k:=1$  to resources.size do
8:         if  $rap(resources(k), processes(i).tasks(j), processes(i).tasks(j).productType) = 1$ 
then
9:            $applicableResource(l) \leftarrow resources(k)$ 
10:           $l \leftarrow l + 1$ 
11:        end if
12:      end for
13:       $mC \leftarrow 1$ 
14:      //mC is the index of resource with minimum execution cost
15:      for  $n:=2$  to applicableResource.size do
16:        if  $c(applicableResource(n)) < applicableResource(mC)$  then
17:           $mC \leftarrow n$ 
18:        end if
19:      end for
20:      //Schedule applicableResource(mC) to tasks(j)
21:       $mC \leftarrow 1$ 
22:    end for
23:  end for
24: end procedure

```

A 3. és 4. eljárások ütemező algoritmusok. A 3. eljárás időre, míg a 4. eljárás a költségre ütemez. Tekintsük az **Algorithm 3** pszeudokódot. A 2.-4. sorokban látható három tároló, amelyekből az első a folyamatokat, a második az összes erőforrást tárolja, a harmadik pedig majd az eljárás során kerül feltöltésre. Az 5. sorban kezdődő *for* ciklusok rendre a folyamatokon, azon belül a tevékenységeken, és azon belül pedig az erőforrásokon haladnak végig. A 8. sorban egy feltétel kiértékelése látható, amelyben az $rap(\dots)$ függvény megvizsgálja, hogy a k . erőforrás képes-e a j . tevékenységet végrehajtani (a benne szereplő terméktípustól függően) az i . folyamaton belül. Amennyiben igen, úgy az *applicableResource* tárolóba bekerül ez az erőforrás. Nagyon fontos kiemelni, hogy az *applicableResource* tárolóban mindig azok az erőforrások tárolódnak, amelyek az aktuálisan vizsgált j . tevékenységet képesek végrehajtani. Ha minden erőforrást megvizsgált az eljárás, akkor a következő

for ciklusban (14. sor) megkeresi azt az erőforrást, amely a legrövidebb idő alatt képes végrehajtani a tevékenységet. A végrehajtási időt a $t()$ függvény segítségével számítja ki. Végül az erőforrást a 19. sorban a tevékenységhez rendeli hozzá, így a j . *task* beütemezésre került. A 4. algoritmus alapjaiban ugyanolyan elven működik, mint a 3., csak ebben az esetben nem időre, hanem költségre ütemez. Valamint a végrehajtási költséget a $c(\dots)$ függvénnyel számítja ki.

3.7. Grafikus modul

A grafikus modul feladata egy olyan felület biztosítása, amellyel kényelmesen nyomon követhetők a rendszerben történő események. Alább a grafikus felület részletesebb bemutatása következik.

3.7.1. Főablak

Workflow Information Center							
File Network Info							
Workflows Resource type Activities Product type Gantt diagram							
Workflow details							
Choose an action	Process ID	Process name	Start time	Max. end time	Action number	Process type	Status
Refresh Load Delete	1	Zöldszámok panaszkezelés	2014.01.01 8:00:00	2014.01.08 16:00:00	10	BasicProcess	NEW

13. ábra

A 13. ábrán látható a rendszerhez tartozó grafikus felület főablaka. A táblázatban minden, a rendszerbe betöltött munkafolyamat megjelenik. A munkafolyamatokról ismert információ azok azonosítója, neve, végrehajtásának kezdete, végrehajtásának befejezése (maximált felső határ), az aktivitások száma, a folyamat típusa (pl. alap, ütemezett stb.) és a státusza. Utóbbi az újonnan érkezett folyamatok esetében "NEW", a végrehajtás alatt álló folyamatok esetében "IN PROCESS" és a végrehajtott folyamatok esetében "COMPLETED".

3.7.2. Erőforrástípusok fül

Workflow Information Center

File Network Info

Workflows Resource type Activities Product type Gantt diagram

Telefon

Resource type details

Basic data

ID: 1

Telefon

Panaszmegadás

Panasz

Start allocation	End allocation	Number
2014.01.01 8:00:00	2014.01.08 16:00:00	10

Details

Time: 100 (sec)

Cost: NULL

Capacity: 1

Probable error: 0.04

Loss of time: NULL

Interruptable: YES

Additional resources

Resource type	Operation mode	Activity	Product type	From	Until
Üzenetrőgzítő	Üzenetrőgzítő	Panaszmegadás	Panasz	2014.01.01 8:00:00	2014.01.08 16:00:00

Resource substitutability

ID	Operation mode	Activity	Product type	Replacement rate
.....				 %
.....				 %

14. ábra

A **14. ábrán** az erőforrástípusokra vonatkozó adatok jelennek meg. Az itt közölt információk mindig egy munkafolyamatra vonatkoznak; arra, amelyet a felhasználó a főablakban kijelölt. A lenyíló listában csak a kijelölt munkafolyamathoz tartozó erőforrástípusok jelennek meg, amely közül választva automatikusan kitöltődnek az ablakban alább látható táblák. A *Basic data* táblában automatikusan megjelenik az erőforrástípus azonosítója, a három lenyíló listába betöltődnek az erőforrástípushoz tartozó működési módok, aktivitások (amelyekhez a rendszer hozzárendelte őket) és a terméktípusok, amelyek az aktivitásokhoz kapcsolódnak. Az alatta látható táblázatban a rendszer megadja, hogy az adott erőforrástípus milyen intervallumokban lesz alkalmazva, és, hogy hány konkrét erőforrást érint ez.

A *Details* táblában az erőforrástípusról néhány további fontos információ érhető el, úgy mint a végrehajtási idő, a végrehajtási költség, kapacitás, meghibásodás va-

3. DÖNTÉSTÁMOGATÓ ALRENDSZER

lőszínűsége, a meghibásodás miatt kieső időtartam, valamint, hogy megszakítható-e az adott tevékenység az adott erőforrástípus mellett.

Az ablak alján látható táblákban rendre az addicionális erőforrástípusok és azok az erőforrástípusok szerepelnek, amelyekkel az aktuális erőforrástípus esetleg helyettesíthető lenne.

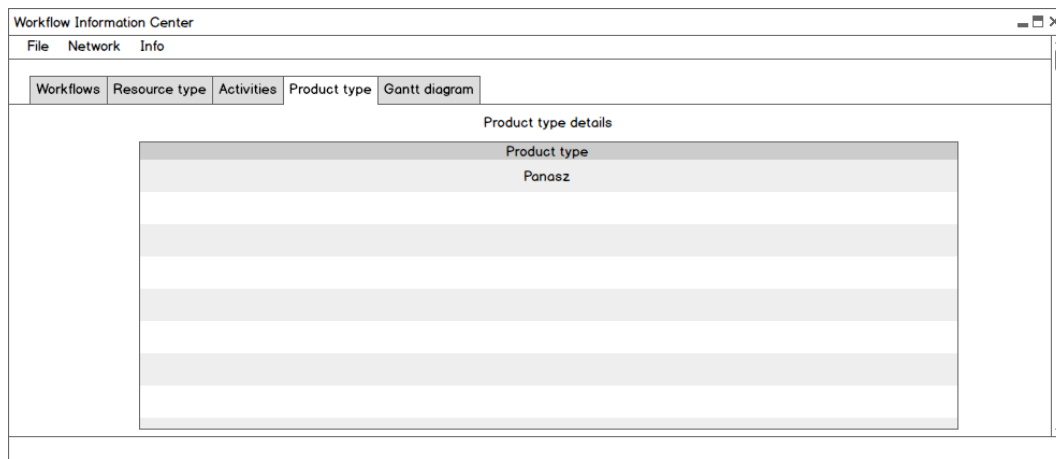
3.7.3. Aktivitások fül

Workflow Information Center		
File Network Info		
Workflows	Resource type	Activities
Product type Gantt diagram		
Activity details		
Activity name	Precondition	Consequence
Zöld számon történő panaszmegadás	NULL	Beérkezett üzenet meghallgatása
Beérkezett üzenet meghallgatása	Zöld számon történő panaszmegadás	Az ügyfél személyes adatainak felvitele a rendszerbe
Az ügyfél személyes adatainak felvitele a rendszerbe	Beérkezett üzenet meghallgatása	A panasz rögzítése a weboldalon
A panasz rögzítése a weboldalon	Az ügyfél személyes adatainak felvitele a rendszerbe	Továbbítás megfelelő ügyintézőnek
Továbbítás megfelelő ügyintézőnek	A panasz rögzítése a weboldalon	Beérkezett anyag elolvasása
Beérkezett anyag elolvasása	Továbbítás megfelelő ügyintézőnek	A panasz feldolgozása
A panasz feldolgozása	Beérkezett anyag elolvasása	A panasz megoldásának bejegyzése
A panasz megoldásának bejegyzése	A panasz feldolgozása	Visszajelzés küldése az ügyfélnek
Visszajelzés küldése az ügyfélnek	A panasz megoldásának bejegyzése	Az ügy zárása
Az ügy zárása	Visszajelzés küldése az ügyfélnek	NULL
Zöld számon történő panaszmegadás		
Resource type	Operation mode	Product type
Telefon	Telefon	Panasz

15. ábra

A **15. ábrán** látható az aktivitások tulajdonságait leíró fül. A felső táblázatban az aktivitások alap adatai jelennek meg, úgy mint a neve, előfeltétele, ráépülő tevékenység. Az alatta látható lenyíló listából azok az aktivitások választhatóak ki, amelyek a főoldalon kijelölt munkafolyamathoz tartoznak. A kiválasztott aktivtáshoz a rendszer megadja, hogy milyen erőforrástípussal hajtható végre, milyen működési módban és milyen terméktípus tartozik hozzá.

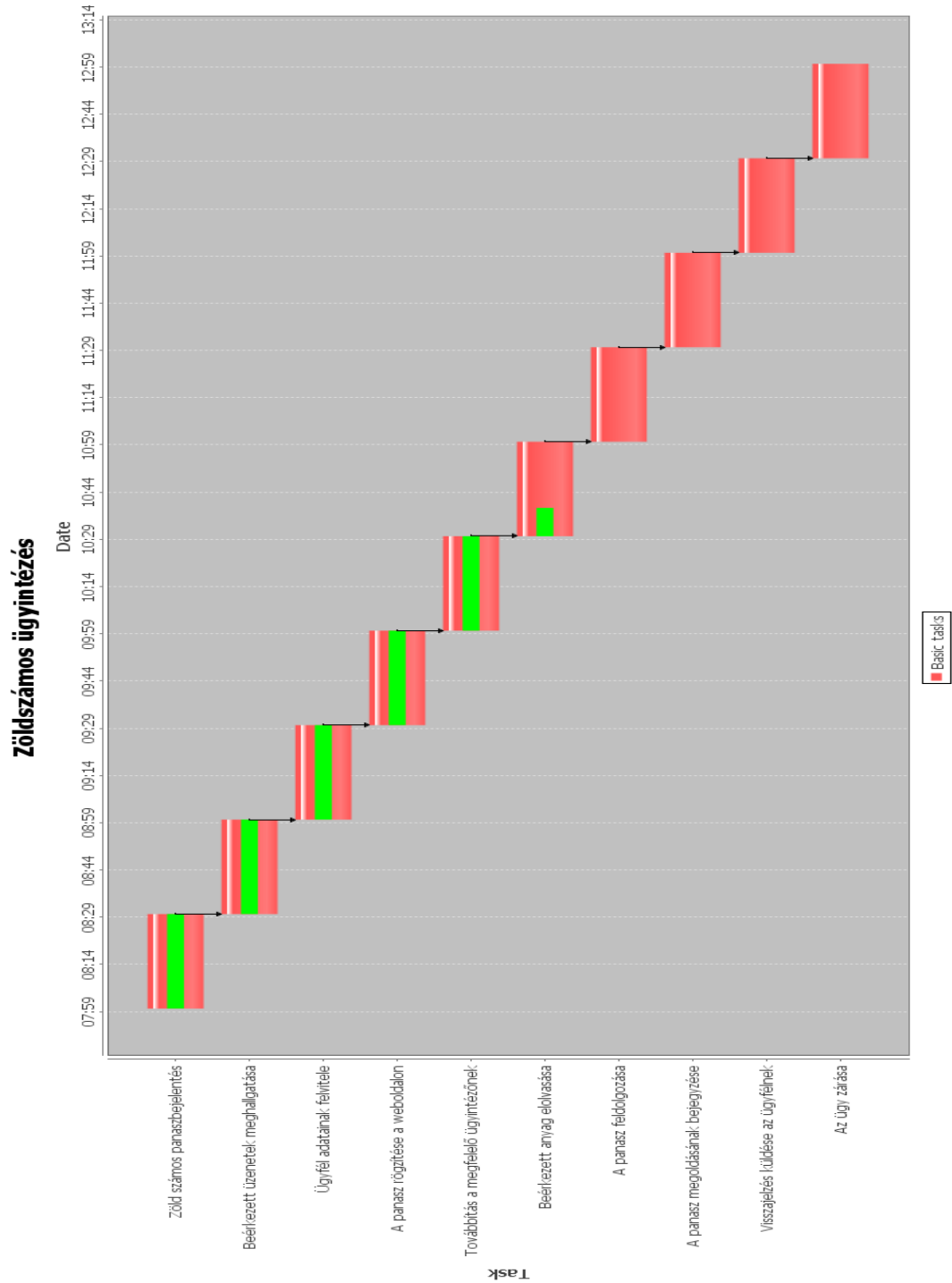
3.7.4. Terméktípusok fül



16. ábra

A **16. ábrán** egy listát jelenít meg a felület, amelyen egy konkrét munkafolyamathoz tartozó összes terméktípus megtalálható.

3.7.5. Gantt-diagram fül



17. ábra

3. DÖNTÉSTÁMOGATÓ ALRENDSZER

A *Gantt diagram* fülön egy gantt diagram jelenik meg, amely a kiválasztott munkafolyamatban szereplő tevékenységeket, a tevékenységek közötti kapcsolatokat fogja ábrázolni. A diagram képes nyomonkövetni azt is, hogy egy konkrét tevékenység mekkora %-ban készült el. A diagramra egy példa a **17. ábrán** látható.

4. Összefoglaló

A döntéstámogató rendszer megtervezésével és megvalósításával a cél egy olyan komplex rendszer kialakítása volt, amely képes az üzleti folyamatok végrehajtásának optimalizálására, a hangsúlyt az erőforrások megfelelő felhasználására helyezve. A munka során elsőként a rendszer felépítését és működését modelleztük. Itt öt fő irányelv szerint készültek el a modellek; adatok feldolgozása, a rendszerkomponensek közötti és a külső rendszerrel való kommunikáció, az erőforrások kezelése, a munkafolyamatok kezelése, ütemezés és szimuláció. Természetesen a modellekben ügyeltünk arra, hogy a rendszer képes legyen az önálló, emberi beavatkozást nem igénylő működésre. A megvalósítás során az egyik legfontosabb feladatunk volt az ütemezés megtervezése, amely a rendszer legfontosabb feladatát, a hatékony erőforráskiosztást valósítja meg. Így kidolgoztunk egy olyan ütemező algoritmust, amely két fontos tényező, az idő és a költség figyelembevételével képes az erőforrásokat a munkafolyamatok tevékenységeihez rendelni.

A döntéstámogató rendszerhez kapcsolódik egy logelemző modul. A rendszer későbbi fejlesztési szakaszaiban szeretnénk úgy megvalósítani a modul működését, hogy képes legyen együttműködni a döntéstámogató rendszerrel és a naplózott adatokból képes legyen olyan információkat kinyerni, amelyekkel a szimuláció még hatékonyabbá, intelligensebbé tehető. Ilyen jellegű adatok lehetnek majd az erőforrásokat jellemző hatékonyság és megbízhatóság. A további terveink között szerepel, hogy a rendszer intelligens módon viselkedjen, amely azt jelenti, hogy az üzleti folyamatokat felügyelő felhasználókkal interaktívan kommunikáljon, és az esetleges, felhasználóktól származó helytelen döntéseket kiszűrje. Utóbbi esetben képes legyen egy másik alternatíva felkínálására és természetesen azt képes legyen megindokolni.

Irodalomjegyzék

- [1] W.M.P. van der Aalst, H.A. Reijers, A.J.M.M. Weijters, B.F. van Dongen, A.K. Alves de Medeiros, M. Song, and H.M.W. Verbeek, Business Process Mining: An Industrial Application, *Information Systems*, 32(5), 2007, pp. 713-732.
- [2] W.M.P. van der Aalst, B.F. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A.J.M.M. Weijters, Workflow Mining: A Survey of Issues and Approaches, *Data and Knowledge Engineering*, 47(2), 2003, pp. 237-267.
- [3] A. Bala, I. Chana, Survey of Various Workflow Scheduling Algorithms in Cloud Environment, 2nd National Conference on Information and Communication Technology (NCICT), 2011, pp. 26-30.
- [4] I. Barba, C. Del Valle, A Job-Shop Scheduling Model of Software Development Planning for Constraint-based Local Search, *Int. Journal of Software Engineering and Its Applications*, Vol.4, No.4, 2010, pp. 1-16.
- [5] B. Brahimi, C. Aubrun, E. Rondeau, Modelling and Simulation of Scheduling Policies Implemented in Ethernet Switch by Using Coloured Petri Nets, In *Proceedings of the 11th IEEE Int. Conf. on Emerging Technologies and Factory Automation*, 2006, pp. 667-674.
- [6] R. Bhaskar, H. S. Lee, A. Levas, and R. Petrakian, F. Tsai, and B. Tulsiek, Analyzing and re-engineering business processes using simulation, In *Simulation Conference Proceedings*, 1994, Winter, IEEE, pp. 1206-1213.
- [7] P. Brucker, *Scheduling algorithms*, 5th ed., Springer, 2007
- [8] T. Dulai, Á. Werner-Stark, Immediate event-aware routing based on cooperative agents, *Proceeding of Factory Automation 2012*, 2012, pp. 144-148.
- [9] J. Ghosh, Distributed task scheduling and human resource distribution in industrial service solution production: a simulation application, *International Journal of Production Research*, 51(10), 2013, pp. 2900-2914.
- [10] A. Gorbenko, V. Popov, Task-resource Scheduling Problem, *International Journal of Automation and Computing*, 9(4), 2012, pp. 429-441.

- [11] F. Gua, L. Yu, S. Tian, and J. Yu, A workflow task scheduling algorithm based on the resources' fuzzy clustering in cloud computing environment, *International Journal of Communication Systems*, 2014.
- [12] M. Merdan, T. Moser, W. Sunindyo, s. Biffl, and P. Vrba, Workflow scheduling using multi-agent systems in a dynamically changing environment, *Journal of Simulation*, 7, 2013, pp. 144-158.
- [13] S. Murthy, R. Akkiraju, J. Rachlin, F. Wu, Agent-based cooperative scheduling, In *Proceeding of AAAI Workshop on Constraints and Agents*, 1997, pp. 112-117.
- [14] M.L. Pinedo, *Scheduling: Theory, Algorithms, and Systems*, 4th ed., Springer, 2012
- [15] A. Rozinat, S. Zickler, M. Veloso, W. M. van der Aalst, and C. McMillen, Analyzing multi-agent activity logs using process mining techniques, In *Distributed Autonomous Robotic Systems*, Springer Berlin Heidelberg, 8, 2009, pp. 251-260.
- [16] D. R. Sule, *Production Planning and Industrial Scheduling: Examples, Case Studies and Applications*, 2nd ed. CRC Press, 2007
- [17] K. Tumay, Business process simulation, In *Proceedings of the 28th conference on Winter simulation*, IEEE Computer Society, 1996, pp. 93-98.
- [18] P. Varalakshmi, A. Ramaswamy, A. Balasubramanian, and P. Vijavkumar, An Optimal Workflow Based Scheduling and Resource Allocation in Cloud, In *Advances in Computing and Communications*, Springer Berlin Heidelberg, 2011, pp. 411-420.
- [19] J. Xu, C. Liu, X. Zhao, S. Yongcareon, Business process scheduling with resource availability constraints, In *Proceeding of OTM'10*, Vol.1, 2010, pp. 419-427.
- [20] Z. Zhao, P. Grosso, J. van der Ham, R. Koning, and C. de Laat, An agent based network resource planner for workflow applications, *Multiagent and Grid Systems*, 7(6), 2011, 187-202.

- [21] <http://www.bpmn.org/>
- [22] <http://jade.tilab.com/wade/>
- [23] <http://jade.tilab.com/>
- [24] **WADE USER GUIDE; Giovanni Caire (Telecom Italia S.p.A); 2010-2011**
- [25] <http://www.json.org>
- [26] **Developing Multi-Agent Systems with JADE; Fabio Bellifemine (Telecom Italia, Italy), Giovanni Caire (Telecom Italia, Italy), Dominic Greenwood (Whitestein Technologies AG, Switzerland); ISBN: 978-0-470-05747-6 (HB); 2007**
- [27] **Diszkrét matematika, Szendrei Ágnes, ISSN 1417-0590, 6. kiadás, 2004**
- [28] http://www.wi-inf.uni-duisburg-essen.de/FGFrank/ecomod/index.php?workflow=found_2&&lang=en