Információszerzés folyamatbányászati módszerekkel a ProM keretrendszer segítségével

A leírás lényege, hogy konkrét példákon keresztül adjon segítséget a folyamatbányászattal ismerkedőknek a ProM keretrendszer felhasználásáról.

1. A keretrendszer ismertetése

Ahhoz, hogy egy eljárást hasznosítani tudjunk, nélkülözhetetlen a megvalósítási környezetének ismerete. A ProM 5.2 vagy 6.9 kiváló választás a folyamatok modellezésére, és tökéletes alapot ad a belőlük kinyerhető adatok értékelésére, felhasználására.

A ProM szabadon letölthető és telepíthető program.

A program a http://www.processmining.org oldalról tölthető le.

A ProM 6 használatát 3 különböző funkcionális nézet határozza meg: a „Workspace” fül, amely a munkafelület nézetet jeleníti meg, az „Action” fül, amely a különböző műveleteket jeleníti meg, és a „View” fül, ami által az elkészített munka tekinthető meg. Minden nézethez más és más funkciók társulnak, annak előre definiált szerepe szerint. Az 1. ábrán a Workspace fül nézete, és annak elemei láthatóak.

1. ábra – A Workspace elemei

2. Példa log-ok elemzése

Ebben a fejezetben példa log-ok segítségével szeretnénk bemutatni, hogy hogyan kell használni a ProM-et és választ adni néhány, a vállalati vezetők által gyakran felmerülő kérdésre. Ezek a kérdések általában az alábbiak szoktak lenni:

- Mi az átlagos/minimum/maximum átvitele ideje az egyes eseteknek?
- Melyik út/utak tartanak túl sokáig? Mik az ezekhez tartozó kritikus al-utak?
- Mi az átlagos szolgálati idő az egyes feladatoknál?
- Mennyi az idő telik el két feladat között a folyamatmodellben?
- Valójában mi történik az esetekkel, hogyan vannak végrehajtva?
- Hogy vannak a szabályok valójában teljesítve?
- Hány ember kapcsolódik egy esethez?
- Mi a kommunikációs struktúra és függőségi viszony az emberek között?
- Mennyi transzfer történik az egyes szabályok között?
- Kik a fontos szereplők a kommunikációs áramlásban?
- Ki kinek dolgozik be?
- Kik dolgoznak azonos feladaton?

Ezek általános jellegű kérdések, melyek mind egy adott folyamat hatékonyabbá tételét szolgálják. Használatukkal többet tudhatunk meg egy adott folyamatról és minőségi, időbeli, financiális előnyökhöz juthatunk. Ismétlésképpen, a folyamatbányászat célja az információk automatikus felfedezése az esemény log-okból. Ezeknek az adatoknak a segítségével olyan új rendszereket lehet üzembe helyezni, amelyek üzleti folyamatok végrehajtását segítik, vagy egyfajta visszajelző eszközként vizsgálatok, elemzések, fejlesztések hatékony részei lehetnek. A folyamatbányászati technikák fő előnye az, hogy az információk elfogulatlanul gyűjthetőek össze, és azt mutatják meg, hogy mi történik egy szervezeten belül egy aktuális eseménnyel kapcsolatban, nem pedig azt, amit az emberek gondolnak, hogy történik.

2.1.A repairExamples.xes log elemzése

Az első dolog, amit egy log vizsgálatakor, bányászatakor tennünk kell, az az, hogy betöltésre kerül a ProM környezetbe. Az első tesztfeladat során, a repairExamples.xes log-ot használjuk.

Egy példa meghatározása is szükséges, melyet vizsgálhatunk: legyen ez egy hibás telefonokat javító vállalat működési folyamatát leíró folyamat. A vállalat 3 különböző telefont

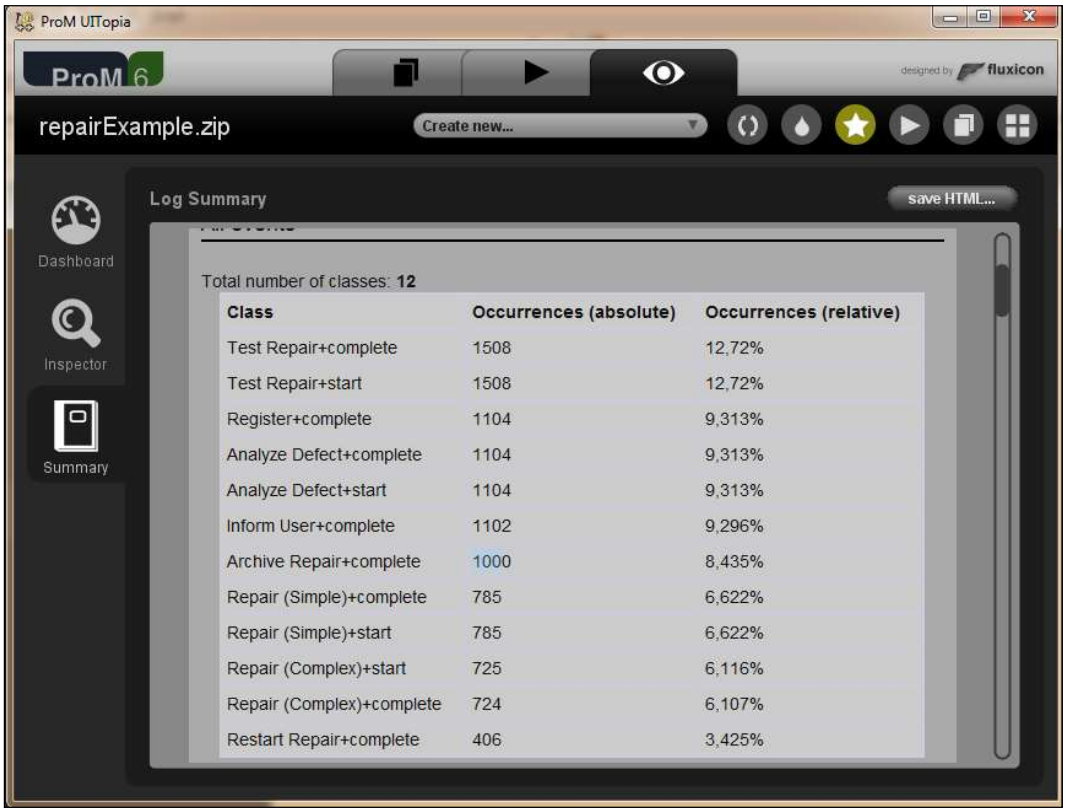
képes javítani: T1, T2 és T3. A folyamat az ügyfelek által küldött hibás készülékek regisztrálásával kezdődik. Ezután az eszköz a probléma felderítését végző osztályra kerül, ahol megvizsgálják és kategorizálják az eszközt a talált hiba alapján. Későbbi megjavítása alapján, az előre definiált összesen 10 különböző hibakategória egyikébe kerül a telefon. Miután a hiba behatárolása megtörtént, az eszközt tovább küldik a javítási részlegre, miközben az ügyfél is kiértesítésre kerül. A javítási részlegnek két csoportja van: az egyik az egyszerű problémákat képes megoldani, a másik pedig a komolyabbakat. Vannak azonban olyan hibák is, melyek megoldását mind a két csoport el tudja látni. A hiba elhárítása után, a telefon a minőségbiztosítási részlegre kerül. Itt ismét ellenőrzésre kerül, hogy a hiba valóban elhárításra került-e. Amennyiben további probléma áll fent, az eszköz újabb vizsgálatra visszakerül a javítási részlegre. Ha pedig a telefon valóban hibamentes, akkor elküldik az ügyfélnek és az esetet lezárják. Azért, hogy időt takarítsanak meg, a vállalat csak korlátozott számban próbál megjavítani egy telefont. Ha a javítás nem sikerül, akkor az ügyfélnek küldenek egy teljesen új telefont, és az esetet lezárják.

Mielőtt elkezdenénk a példa folyamatbányászati ismertetését, rendkívül fontos tisztázni még azt is, hogy legyen elképzelésünk arról, hogy milyen adatokra, információkra van szükségünk egy esemény log-jaiból. Ennek fő oka az, hogy csak akkor válaszolhatunk konkrét kérdésekre, ha a válaszhoz szükséges információt a log tartalmazza is. Például abszolút nem lehet számolni egy eset átfutási idejével, ha a hozzá tartozó log nem tartalmaz információt az egyes feladatok elvégzésének idejéről. Valamint az is nagyon fontos, hogy képesek legyünk a szükségtelen információkat kiszűrni, és „megtisztítani” a log-ot a felesleges adatoktól. Például amikor egy konkrét esetben csak az elvégzett feladatokra vagyunk kíváncsiak; értelemszerű, hogy ebben az esetben szükségtelen a folyamatban lévő feladatok pontos számának ismerete. Ennek egy további vetülete, hogy azt is tudnunk kell, hogyan kell kezelni ezeket a megtisztított log-okat, hogy elkerülhessük a felesleges munkát.

Visszatérve a példához, meghatároztunk néhány általános érvényű kérdést, amikre a log elemzésével majd választ tudunk adni:

- Hány eset/folyamat van a log-ban?
- Hány feladat van a log-ban?
- Mennyi erőforrás van a log-ban?
- Van futó eset a log-ban?
- Melyik erőforrás melyik feladathoz van rendelve?

Erre az öt kérdésre a log összefoglalójának megtekintése után kaphatunk választ. Ehhez a Workspace ablakon belül a Visualize menü megnyitására van szükség, ahol a Summary alatt válik láthatóvá a számunkra fontos összegzés. Az MXML Legacy Classifier / End events tartalmazza az első négy kérdésre a választ, hiszen ebből a részből világosan látszik, hogy jelenleg 1000 lezárt eset és 1104 folyamatban lévő van a vállalatnál. Az 2. ábra is jól mutatja, a vállalat egyes feladatrészeinek összegzését.



Class	Occurrences (absolute)	Occurrences (relative)
Test Repair+complete	1508	12,72%
Test Repair+start	1508	12,72%
Register+complete	1104	9,313%
Analyze Defect+complete	1104	9,313%
Analyze Defect+start	1104	9,313%
Inform User+complete	1102	9,296%
Archive Repair+complete	1000	8,435%
Repair (Simple)+complete	785	6,622%
Repair (Simple)+start	785	6,622%
Repair (Complex)+start	725	6,116%
Repair (Complex)+complete	724	6,107%
Restart Repair+complete	406	3,425%

2. ábra – A hibadetektálási log adatai

Az ötödik kérdésre a választ az Event name AND Resource rész adja meg: ebből jól látszik, hogy a SolverC kezdetű azonosítóval rendelkező emberek foglalkoznak a bonyolult feladatokkal, míg a SolverS kezdetű azonosítóval ellátott emberek pedig a könnyebb problémákkal. Az is megfigyelhető, hogy mindkét csoportban 3-3 ember dolgozik. Az Inspector menüpontban az egyes esetek részletesen elemezhetőek, kiértékelhetőek.

2.2. Az exercise1.xes log elemzése

Ebben a példában a ProM további felhasználási lehetőségeit szeretnénk megmutatni. A felhasznált log az exercise1.xes, melynek a már ismert módon való betöltése után lehetőségünk van a folyamatbányászat által értékes információkat kinyerni. Az esemény log-ja az alábbi trace-eket tartalmazza:

1x Case1 A B C D

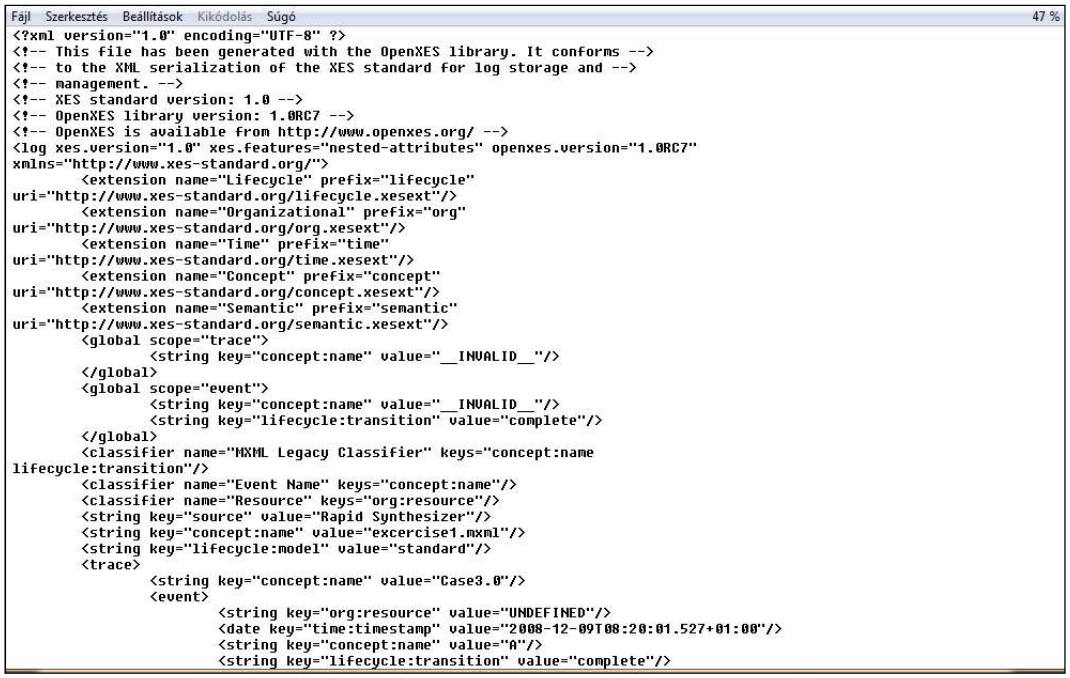
1x Case2 A C B D
1x Case3 A E D

A ProM Visualization menüjének segítségével lehet megválaszolni az alábbiakban feltett kérdéseket:

- Milyen időintervallumok között fordulnak elő események a log-ban?
- Mikor és ki által hajtódik létre az A esemény Case3 esetében?
- Mi a relatív előfordulása és a gyakorisága a D eseménynek?

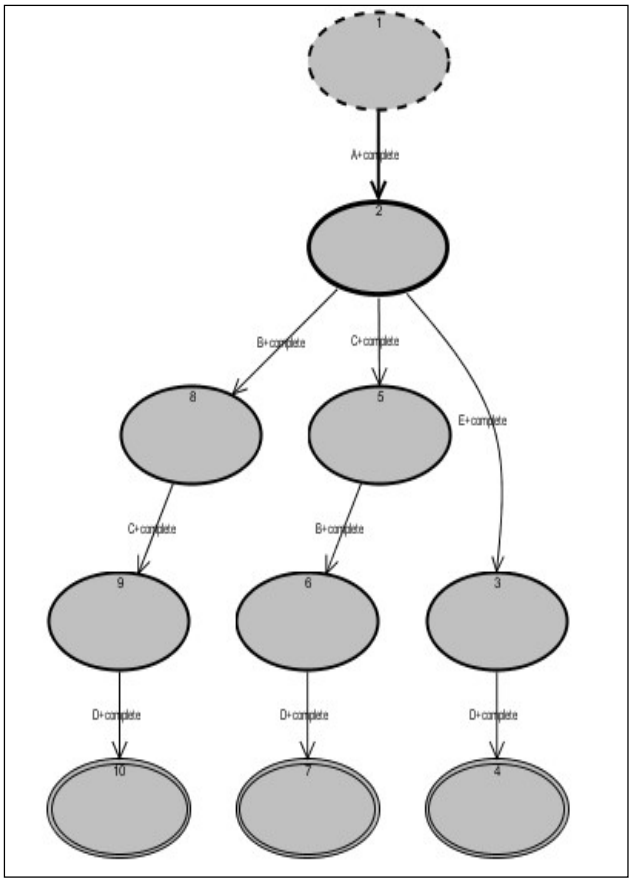
Az első kérdésre a választ az exercise1.xes log betöltése után, a Visualization menü Dashboard pontjában találhatjuk meg. A log info felirat alatt látható az események előfordulásának időintervalluma: 2008. dec. 09. kedd 08:20:01 – 2008.dec.09. kedd 08:23:01, azaz 3 perces időkeret. Ez az elsőre nem sokat „érőnek" tűnő információ valójában sokkal többet rejt magában, mint hinnénk. Hiszen ez a folyamatbányászat alapja, annak virtuális időkeretét adja meg. Ez a legfontosabb kezdeti információ, ami nélkülözhetetlen a további adatok kinyeréséhez, illetve annak behatárolására, hogy mely időszakból szeretnénk kinyerni a kívánt információkat. A második kérdésre a választ az Inspector menüben az Instances részen kiválasztható Case 3.0 megjelenítésével kapjuk meg: a Case 3.0-hoz 3 esemény tartozik, és a D esemény 2008. dec. 09. 8:22:01.527-kor történt meg és nem lett meghatározva, hogy ki által lett végrehajtva. A harmadik kérdésre a Summary menüpontban kapunk választ, mely tartalmazza az összes eseményt is: a D esemény relatív előfordulása 100%. Az A esemény gyakoriságát az Inspector menü Explorer pontjában lehet megtekinteni a Case 3.0 rész megnyitásával: 100%, ami azt jelenti, hogy ez mindig végrehajtódik.

Ezeknek a kérdéseknek a megválaszolása után, jogosan merül fel bennünk, hogy vajon mit is tartalmaz exercise1.xes log. Ennek megismeréséhez, a log megnyitására van szükség egy text editorban. Az 3. ábrán az exercise1.xes log XML szintakszisa látható. A XES standard további megismeréséhez jó kiindulási alap lehet a http://www.xes-standard.org/ oldal. Az Extensible Event Stream (XES), szabad fordításban bővíthető esemény folyam. A XES tulajdonképpen egy XML-alapú szabvány az esemény log-ok létrehozására. Célja egy olyan általánosan elismert és használt forma, amely képes átmenetet képezni az eszközök és az alkalmazási területek között. Elsődleges célja a folyamatbányászati felhasználása. Legfontosabb tulajdonságai: egyszerűség, rugalmasság, nyújthatóság és expresszivitás.



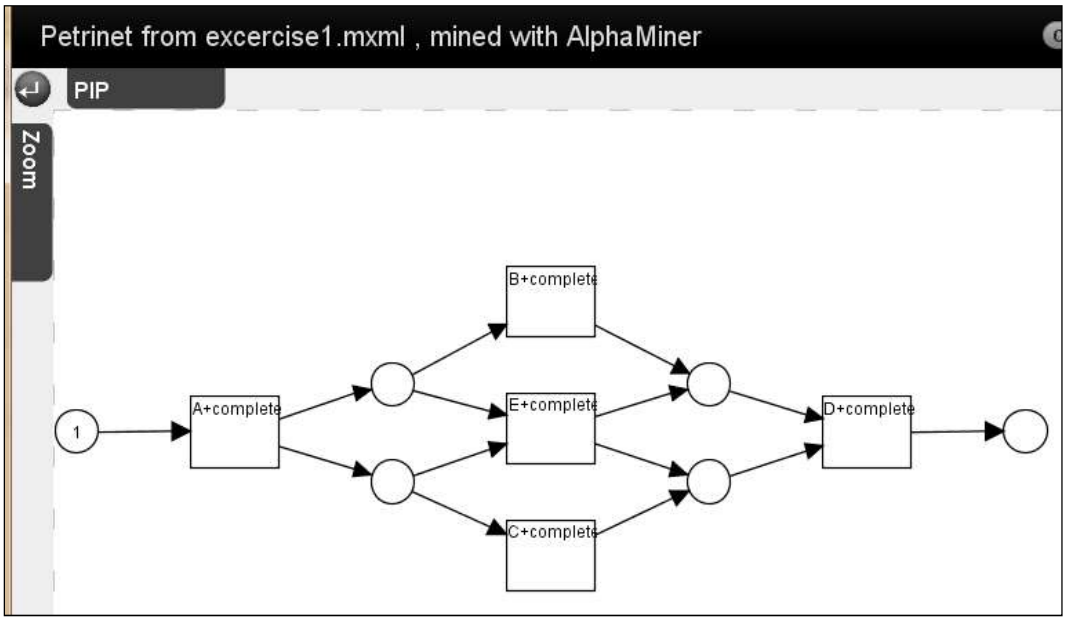
3. ábra – Az exercise1.xes log XML szintakszisa

Következő lépésként próbáljunk meg kézzel előállítani egy Petri-hálót a log felhasználásával. Ezzel a ProM egy újabb, és talán egyik legfontosabb tulajdonságát mutatjuk be, azt, hogy miért kiemelkedő adottságokkal ellátott eszköz a ProM. Az exercise1.xes log és egy Action, azaz művelet segítségével kaptuk meg a 1.10 ábrát. Ezen a log-ban szereplő események közötti átmenetek szerepelnek, a kezdeti állapottal, súlyokkal és elfogadó helyekkel. Az 4. ábra tömören foglalja össze az eseményeket a kezdetitől (A) a végállapotig (D). A képen jól látszanak a végrehajtási utak is: 1-2-3-4; 1-2-5-6-7; 1-2-8-9-10. Ezeken az utakon lehet eljutni a kezdeti állapotból a végállapotokba.



4. ábra – Az exercise1.xes log kimenete

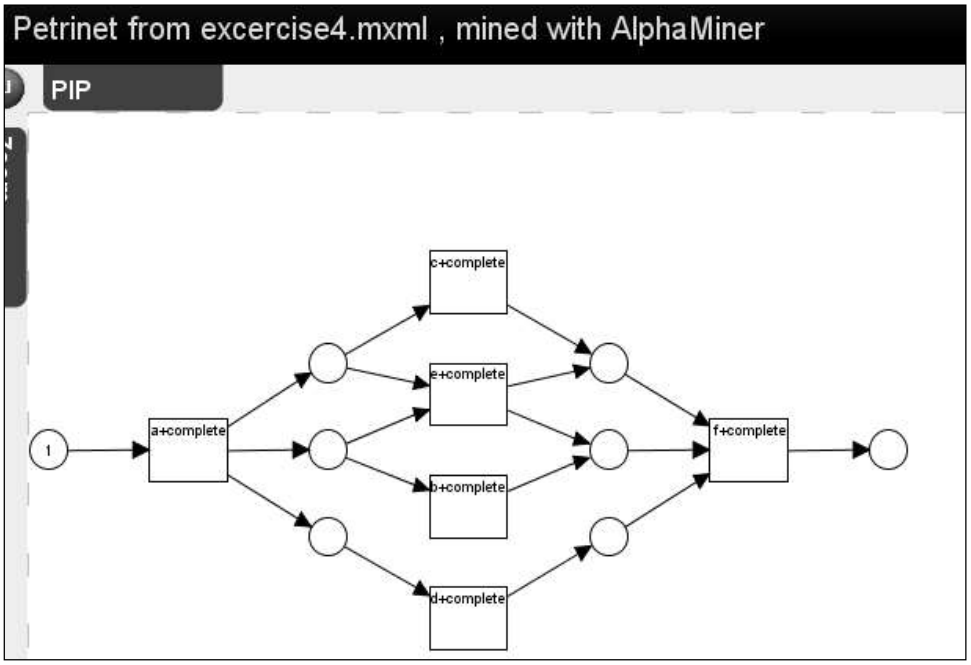
Ezután az AlphaMiner segítségével létrehoztuk a log Petri-hálóját. Ennek az eredménye látható az 5. ábrán.



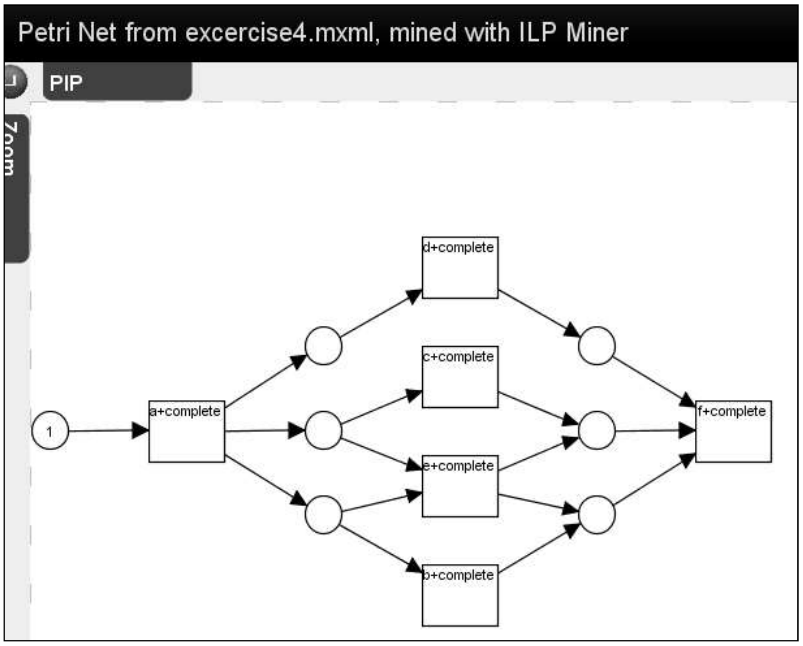
5. ábra – Az exercise1.xes log AlphaMiner által készített Petri-hálója

2.3. Az exercise4.xes log elemzése

Ebben a feladatban az exercise4.xes log-on végeztünk el folyamatbányászati vizsgálatokat az AlphaMiner és az ILP algoritmusok segítségével. A két eredmény összehasonlítása látható a 6. és a 7. ábrákon. Ezeken jól látszik, hogy bár különböző átmeneteket használtunk, az eredmény mégis megegyezik. Ennek az az oka, hogy mindkét algoritmus minden eseményt csak egyszer enged megjeleníteni. Ebből is jól látszik, hogy különböző algoritmusok használata azonos eredményre vezethet.



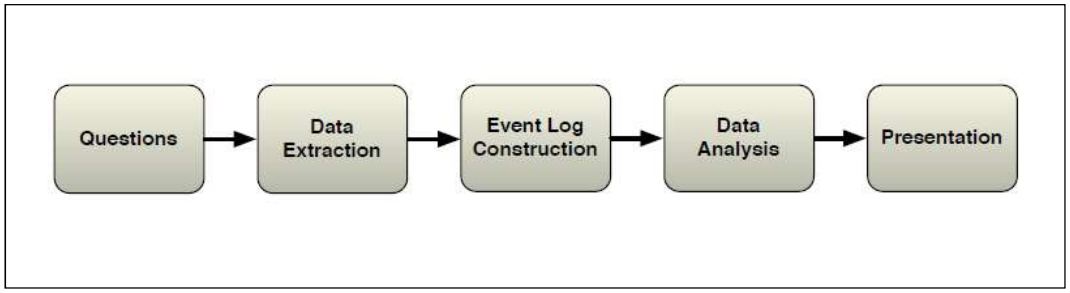
6. ábra – Az exercise4.xes log AlphaMiner használata után



7. ábra - Az exercise4.xes log ILP használata után

3. Folyamatbányászati módszerek vizsgálata működő vállalat munkafolyamat-rögzítő szoftveréből kinyert adatok példáján keresztül

A feladat lényege, hogy egy, a valós életből merített példán keresztül mutassuk be a folyamatbányászat használatának erősségeit és a benne rejlő lehetőségeket. A folyamatbányászat menete előre definiált:



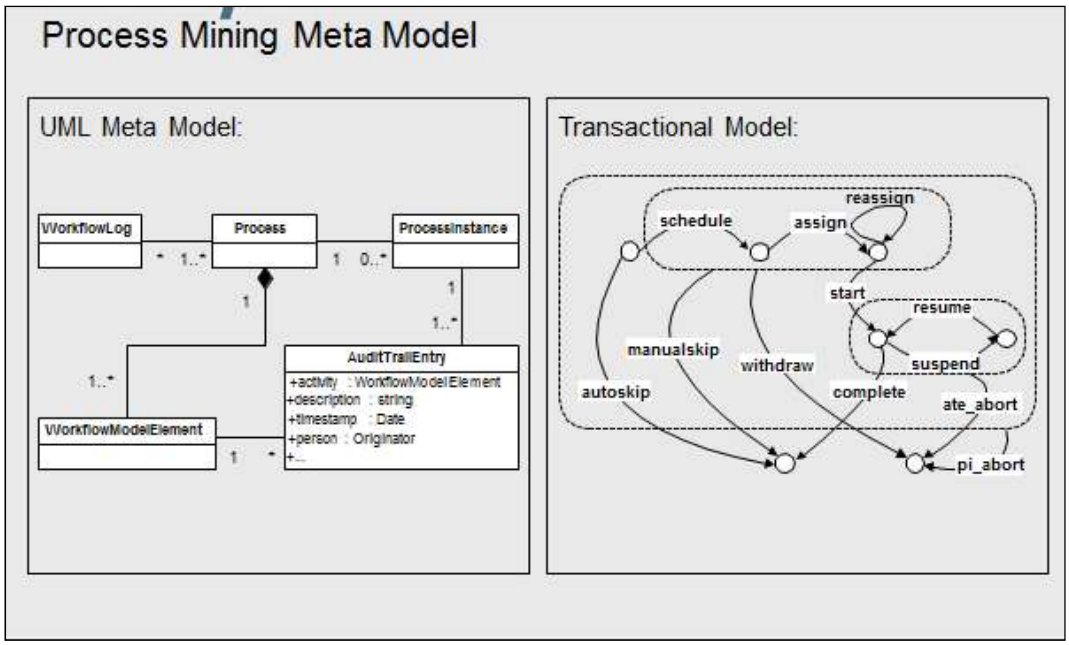
8. ábra – A folyamatbányászat menete

Először fel kell tenni a kérdés(eke)t, amely(ek)re választ várunk a munka végeztével, hogy tulajdonképpen mit is szeretnénk megtudni egy folyamatról. Ebben az esetben néhány alapvető kérdésre szeretnék választ kapni:

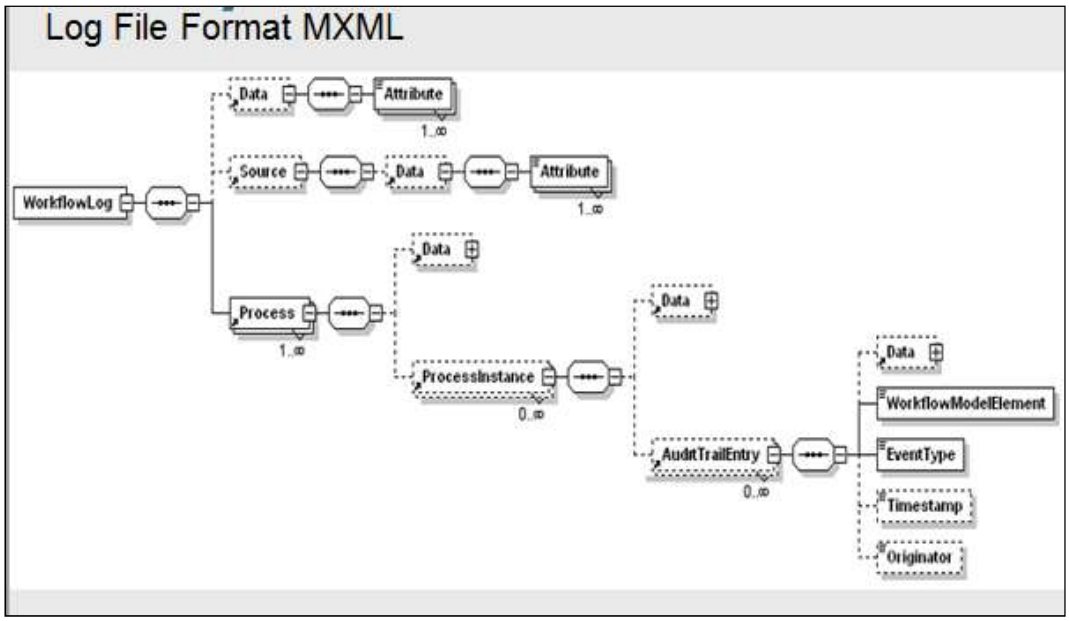
- ki végzi a legtöbb munkát,
- melyik ügyféltől jön a legtöbb munka,
- mi a pontos menete a munkafolyamatnak,
- statisztikai kimutatásokhoz az adatpontok kinyerése.

A második lépés az adatok összegyűjtése, kinyerése. Természetesen a vállalatok nem szívesen osztják meg pontos adataikat ilyen célra, ha ki is adnak adathalmazokat, akkor azok szűrve, „kozmetikázva” vannak. Fontos tisztázni, hogy a feltett kérdésekre csak akkor kaphatunk választ, ha olyan adathalmazzal van lehetőség dolgozni, amely tartalmazza is ezekre a kérdésekre a választ. Számunkra is fontos volt, hogy a kapott adathalmazból csak a számunkra lényegesek legyenek felhasználva, figyelembe véve.

A harmadik lépés az események log-jának elkészítése, egy adathalmazként való kezelése megfelelő formátummal ellátva. Ennek legjobb formája az ún. MXML formátum. A folyamatbányászat meta modellje a 8. ábrán, a log fájl általános mxml ábrája pedig a 9. ábrán látható.



9. ábra – A Process Mining Meta Model



10. ábra – MXML log file forma

A folyamatbányászat következő lépése a kinyert adatok elemzése, értelmezése, majd utolsó lépésként lehet őket prezentálni.

Esetünkben a kiindulási pont egy működő vállalat, ahol speciálisan erre a feladatra kifejlesztett szoftver segítségével próbálják hatékonyabbá tenni a munkavégzést. Az eljárás röviden a következő: az ügyfél e-mailben, telefonon, közösségi üzenetváltó program segítségével, esetleg más módon információt, segítséget vár egy adott témával kapcsolatban. Ezeket a „megkereséseket” nevezzük most leadott munkának (Incomingtask). A leadott munkákat egy kijelölt személy figyeli, és azokat bizonyos kritériumok figyelembe vétele

alapján, az általa megfelelőnek ítélt embernek delegálja. A beérkezés után egy úgynevezett Case ID-val ellátott eseményt nyit, mely ID a munka befejezéséig az aktuális munkához lesz társítva. Általában ezeket a tulajdonságokat kell mérlegelni a helyes delegálás végrehajtásához, természetesen általánosítva a teljesség igénye nélkül: milyen sürgős az adott munka, foglalkozott-e már vele valaki, ki az a személy, aki meg tudja oldani a problémát tudása alapján, igényel-e valós munkavégzést ez a megkeresés, van-e szabad kapacitású ember, akinek lehet továbbítani az igényt. Legjobb esetben az ezeknek a kritériumoknak megfelelő emberhez lesz delegálva a feladat, aki elkezdi annak megoldását. Ennek folyamata általában a hiba értelmezésével kezdődik, majd a rendelkezésre álló tudás alapján el kell hárítani azt, ezt a folyamatot nevezzük Handlingtask-nak. Amikor valakihez delegálva lesz egy adott hiba, akkor egy esemény (Case ID) kerül megnyitásra a munkafolyamat rögzítő programban. A regisztrálása során az alábbi adatok kerülnek rögzítésre:

- Case ID, a beérkező munka sorszáma,
- Operation, mely a Case ID-hoz tartozó aktivitásokat tartalmazza: Incomingtask, Handlingtask, Pendingtask, Closingtask,
- Timestamp: az aktuális aktivitás lekezelésének pontos időpontja, a formátum a következő: DD:MM:YYYY HH:MM:SS,
- Via, azaz a munka milyen forrásból származik: telephone, viafax, viamail, other,
- Customer ID: az ügyfél neve, lehet Customer 1-2-3,
- Complexity: a bejövő feladat komplexitása lehet simplex, medium, complex,
- Servicetype: a munka jellege, amivel kapcsolatos: audit, rollover, incident, KCO, changerequest, Serverboarding, scan, taskid, CSVfiles, update, ownership, backup, upload, validation, Reconcheck,access,database, monitoring, accessrequest, Change, Orphan,client, Shared, error, QEV/CBN, application, Serversunset,
- Originator, az a személy, aki foglalkozott a munkával.

A rögzített esemény a munkavégző (Originator) nevére kerül. Ez az esemény képezi a későbbi log keretét, tartalmazza a legfontosabb kiindulási információkat. Pontos időbélyeggel kell ellátni minden event-et, és így nyomon követhetővé válik egy hiba elhárításának menete. Ez az ún. logolás alapja.

A használható log készítéséhez szükség van néhány fontos szabály betartására:

- egyértelmű megnevezéssel ellátott keret,
- mindenkire érvényes szabványosítás,
- kötelező adatrögzítés,

- riportok rendszeres készítése.

Az első pontnak azért van jelentősége, mert ha bonyolult, csak a fejlesztő által ismert megnevezéseket használunk az adatok megadásának fejlécében, akkor a későbbiekben, a log-ban valószínűleg nem megfelelő adatok fognak megjelenni, vagy olyanok, amelyek nem lesznek hasznunkra, mert a kitöltőjük nem fogja érteni, hogy mi az a pontos és lényeges adat, amit tulajdonképpen meg kell adnia. A második pont talán a legfontosabb: tisztázni kell a felhasználókkal, hogy mindenkire ugyanazok az egységes szabályok vonatkoznak, azok közül a lehetőségek közül válasszanak, amelyeket a program felkínál, és ne kezdjenek el maguktól új elnevezéseket kitalálni. Ettől marad egységes és áttekinthető, könnyen kezelhető a log. A harmadik pont szintén rendkívül lényeges, hiszen ha csak néhányan rögzítik munkájukat, vagy mindenki, de csak alkalmanként, akkor a végső log szintén nem lesz hiteles, abból a rendszerre vonatkozó tényleges adatokat nem lehet majd összesíteni. A kulcsszó ebben az esetben a részinformáció, ami nem fogja a munka egészét lefedni. Ez a módszer gyakran használatos a kimutatások „kozmetikázásához”, a mutatós eredmények kimutatására a valósak helyett. A riportok rendszeres készítése, a teljes összegzés is nélkülözhetetlen, hiszen ezekből is fontos, az aktuális állapotra vonatkozó információkat tudhatunk meg.

3.1. A log készítés alapjai

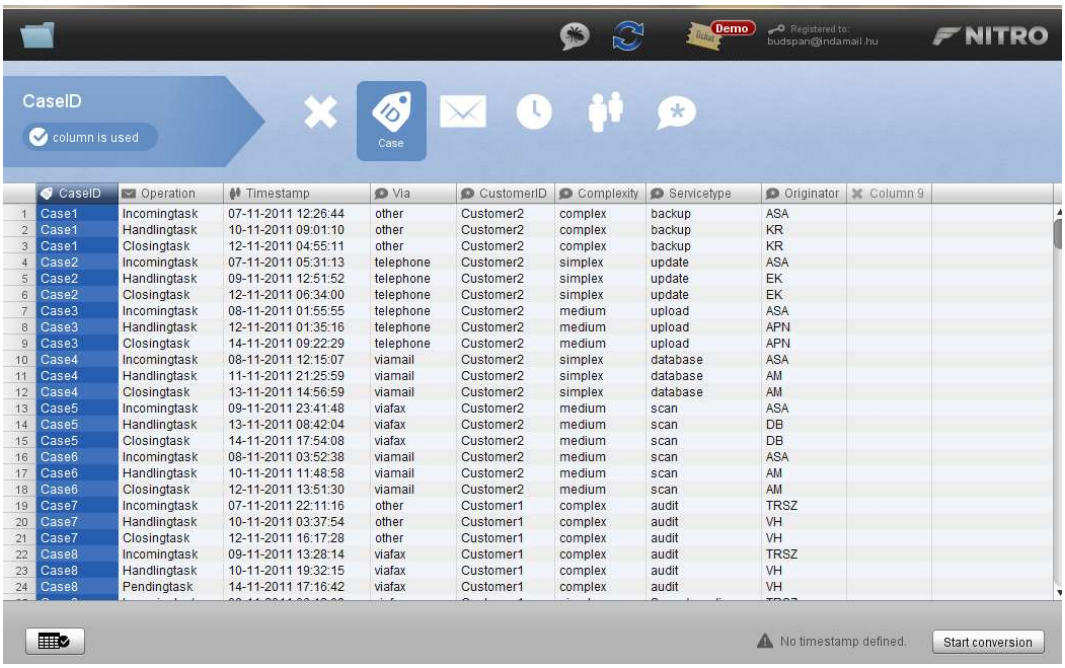
Az első lépés tehát egy log felhasználása előtt, annak áttekintése, az esetleges formai hibák kijavítása. Az egyik leggyakrabban használt forma, melyet szinte minden program fel tud dolgozni az ún. CSV fájl. Ez egy olyan egyszerű fájlformátum, amelyet nagy hatékonysággal lehet táblázati adatok elhelyezésére használni. Fontos része a fejléce, azaz az első sor, ahol a CSV fájlban szereplő adatok sorrendje és tartalma van felsorolva. Ezek betartása és figyelembe vétele a CSV fájl használatának alapja. Sorokból tevődik össze, melyben az aktuális adatok valamilyen írásjellel vannak elválasztva, a könnyebbség kedvéért általában vesszővel, vagy pontosvesszővel. Mivel nagyon egyszerű, ezért kiválóan használható különböző programok közötti információk cseréjére. Legkönnyebben Microsoft Excel segítségével állítható elő. A 11. képen az általunk készített és a későbbiekben felhasznált CSV fájl egy részlete látható.

11. ábra – A felhasznált CSV fájl egy részlete

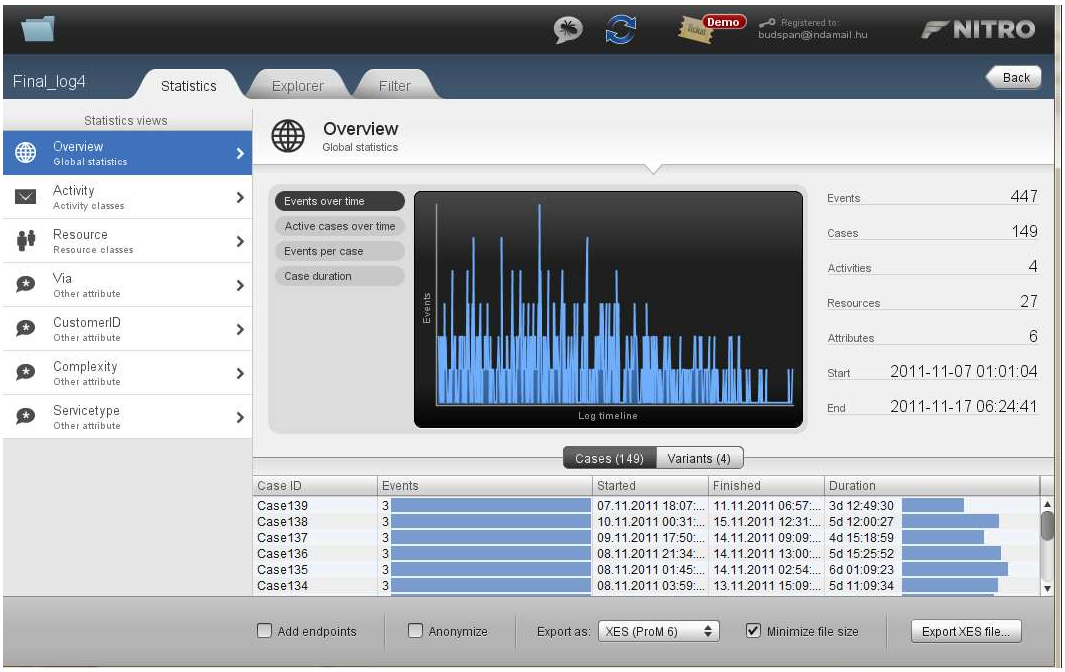
Ahhoz, hogy ezt a CSV fájlt fel tudjuk használni a ProM -ben, szükség van annak átalakítására egy speciális, ún. MXML formátumba. Ennek végrehajtásához szükség volt egy köztes program, a Nitro használatára. A Nitro Fluxicon cég terméke, egy gyors és hatékony eszköz a folyamatbányászathoz, a http://www.fluxicon.com/ oldalon érhető el. A Nitro letöltése után, a Demo verzió volt elérhető számunkra. Ebbe töltöttük be az általunk készített CSV fájlt, és a megfelelő oszlopokat adtuk a megfelelő értékekhez:

- Case = Case ID,
- Activity = Operation,
- Timestamp = Timestamp,
- Via, CustomerID, Complexity, Servicetype = Other,
- Originator = Resource.

A 12. és a 13. ábrákon a Nitro beállításai és a kiértékelt adatok láthatóak.

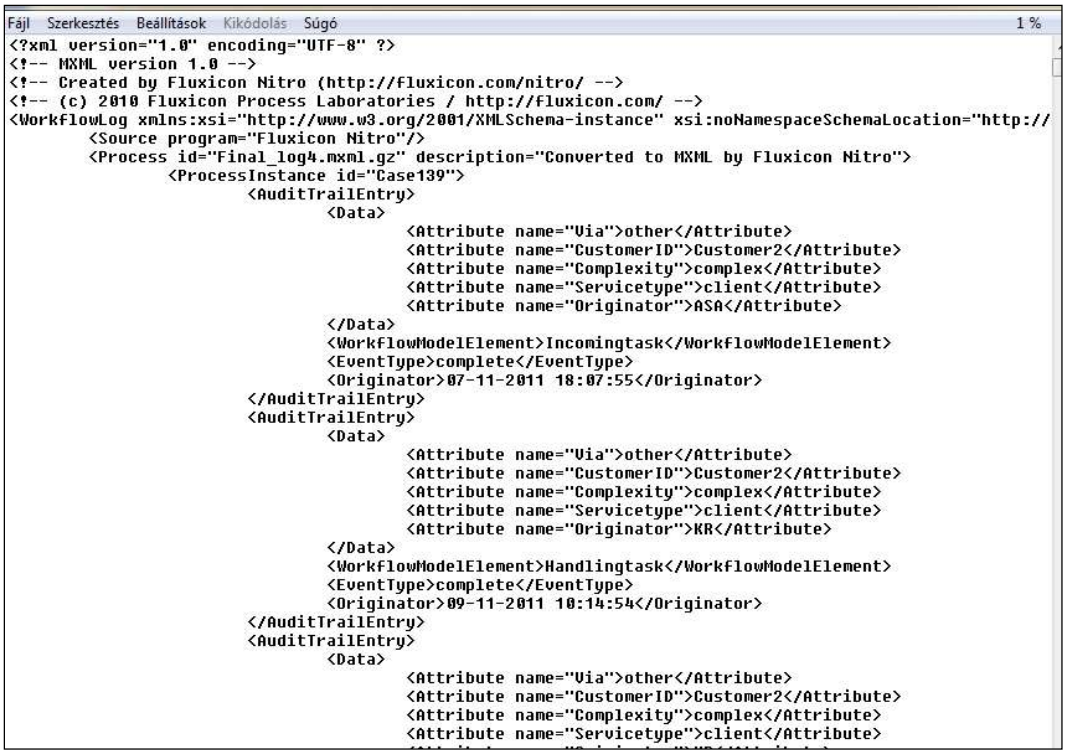


12. ábra – A Nitro beállítása



13. ábra – A feldolgozott adatok megjelenítése

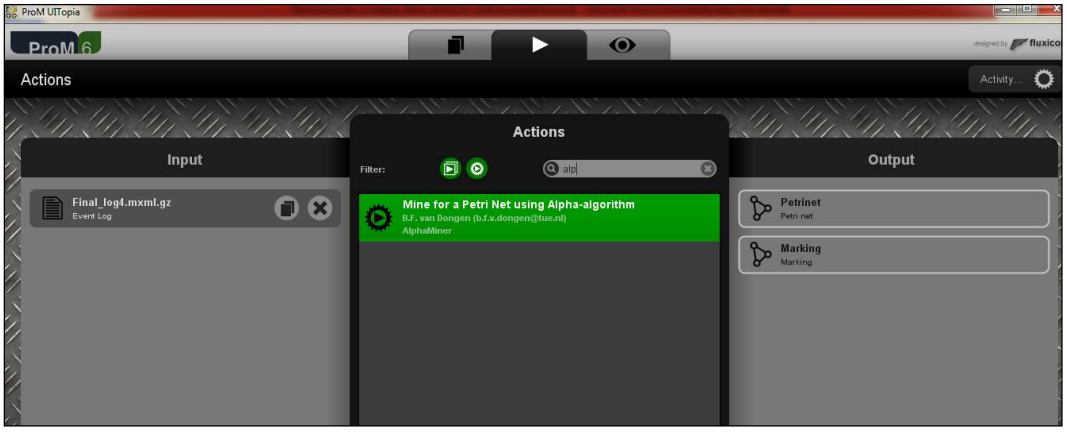
A Nitroban lehetőség van MXML és XES fájlként is lementeni az adatokat. A 14. ábrán a kiértékelt MXML fájl egy részlete látható.



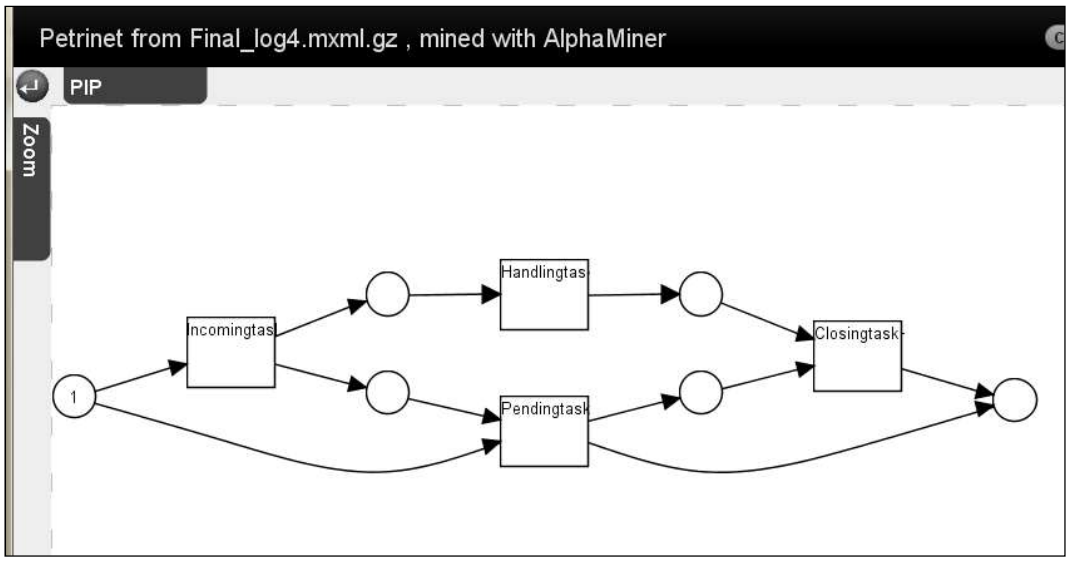
14. ábra – A kiértékelt MXML fájl egy részlete

3.2. A log kiértékelése

Az előzőekben megkapott eredményeket ebben a formátumban már be lehet tölteni a ProM keretrendszerbe. A betöltés után az előre meghatározott algoritmusok közül az α-algoritmust használtuk a 15. ábra szerint, mely felfedi a log alapján a benne rejlő folyamatokat és egy Petri-hálóban jeleníti meg az 16. ábrán látható módon.



15. ábra – A log elemzése α-algoritmussal



16. ábra – A log Petri-hálója

A Petri-háló kirajzolja, hogy valóban 4 különböző Operation van a folyamatban, azok útja is egyértelmű. Az Incomingtask állapotból kétféle állapotba van lehetőség lépni, a Handlingtask és a Pendingtask következhet. Ezekből elvileg csak a Closingtask állapoton lehetne tovább lépni, de a modell szerint van olyan eset is, amikor a Pendingtask-nak nincs lezárása, azaz a folyamatban valamilyen hiba van.

További eredmények is kiolvashatóak a ProM View Resource menüjében. A Dashboard menüpontban jelennek meg a 17. ábrán látható, log-gal kapcsolatos főbb adatok:

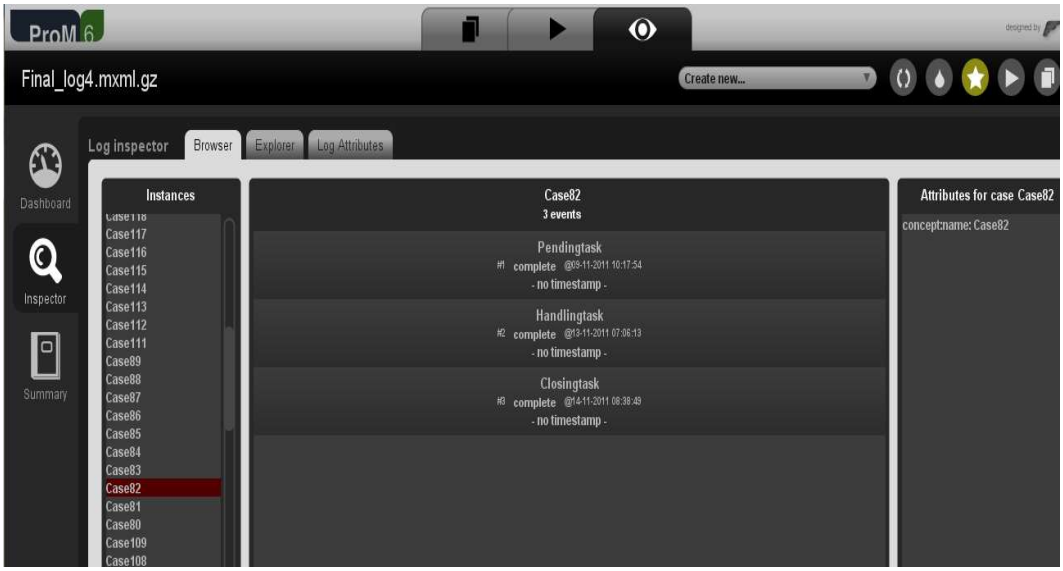
- Cases = 149, a Case ID-k száma,
- Events = 447, az Operation-ok száma,
- Event classes = 4, a különböző típusú Operation.

17. ábra – A folyamatok összegzése

Az Inspector/Explorer menüben van lehetőség az egyes Case megtekintésére. 3 esetben is a megfelelőtől (zöld színnel jelzett) eltérő Case-eket mutat a ProM: Case 8, Case61 és Case82. Ezekben az esetekben piros színnel jelzi a folyamat egy elemét a kis háromszögek közül. A továbbiakban ezek közül vizsgálom meg a Case82-t, hogy milyen hibát fedett fel a ProM. A 18. ábrán láthatóak a megfelelő és a hibás működést jelező esetek egy részlete.

18. ábra – A hibás Case82

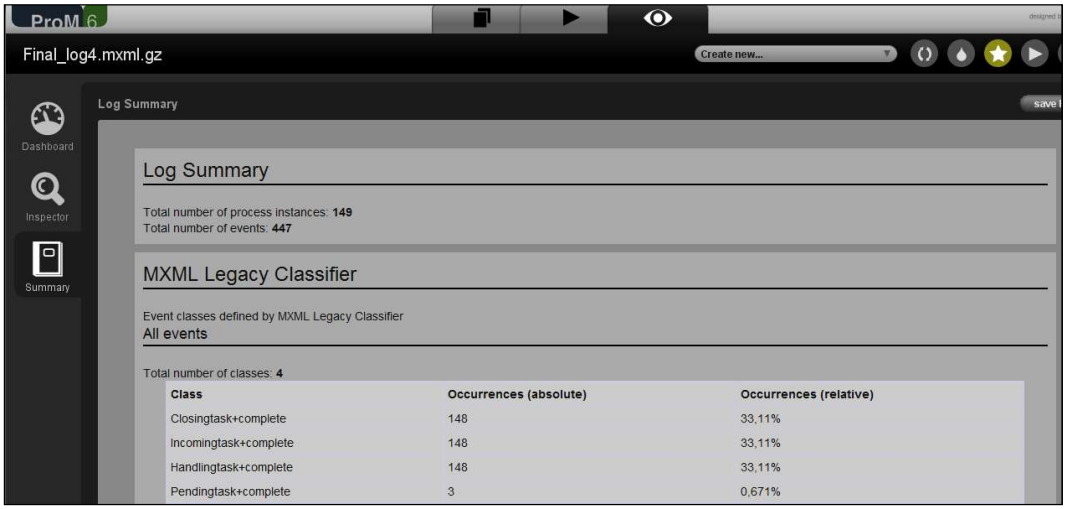
Az 19. ábrán látható a Case82 folyamat részletes kielemzése. A program megmutatja a folyamathoz tartozó részleteket és a hozzájuk tartozó időpontokat.



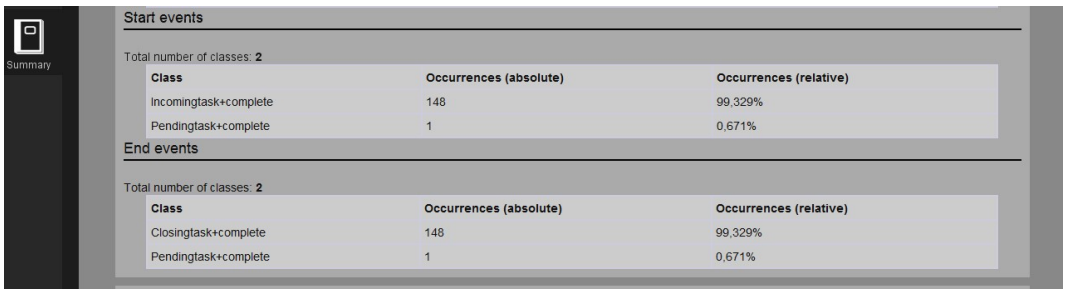
19. ábra – A Case82 állapotai

Jól látszik, hogy a ProM azért jelölte meg hibásként ezt a Case-t, mert a folyamat nem a szokásos Incomingtask állapottal kezdődik, hanem a hibásan kezdeti állapotként beállított Pendingtask állapottal. Ez természetesen hibás működés, amennyiben azt állítjuk, hogy az adott intervallumon (2011-11-07 01:01:04 és 2011-11-17 06:24:41) belül van minden Case kezdeti és záró állapota is. Abban az esetben lenne csak megfelelő, ha ennek az esetnek az Incomingtask állapota még a vizsgálati időszak kezdete előtti időszakra nyúlna vissza, és ezért már csak a Pengingtask esetet tudjuk elemezni.

A 20. és 21. ábrákon az egyes folyamatok összegezve vannak felsorolva, pontosan számszerűsítve.



20. ábra – Log summary 1



21. ábra – Log summary 2