Garázskapú működése

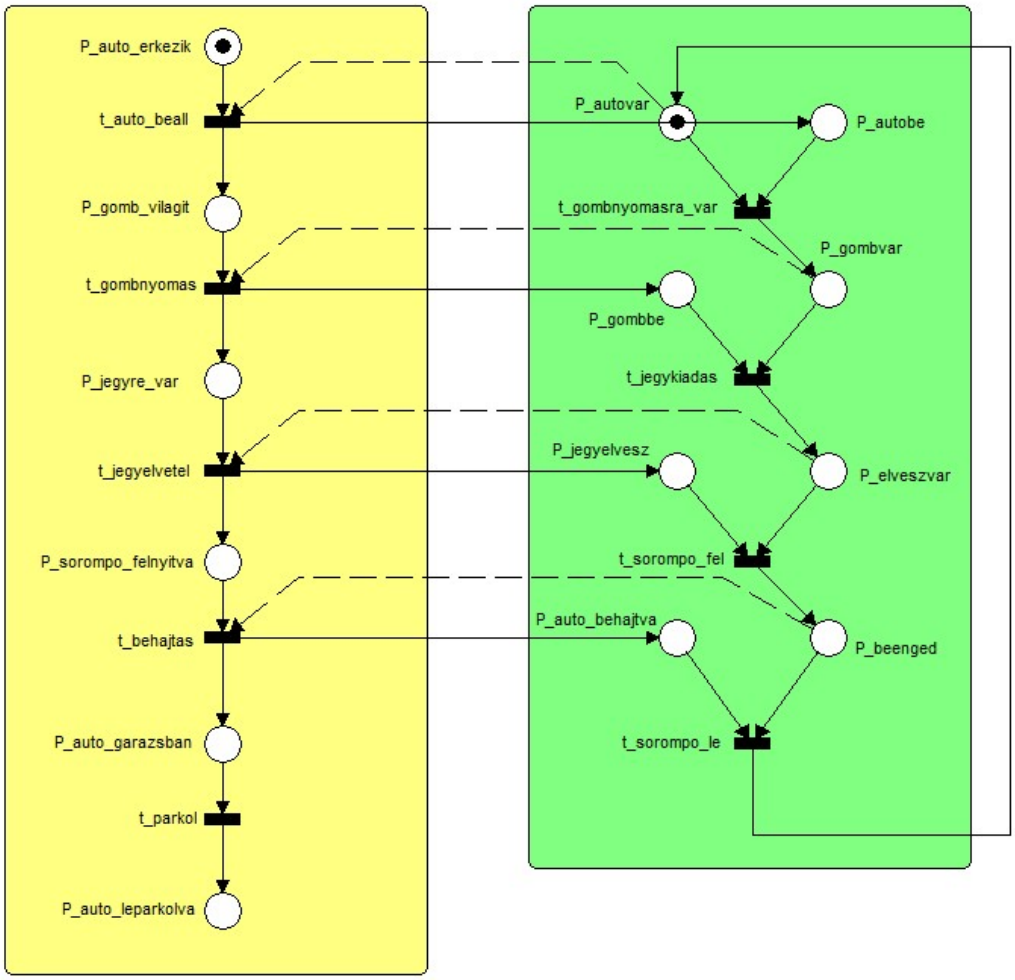
Werner Ágnes (2020)

1. Operátori eljárással irányított garázskapu

Kiindulásként az operátori eljárással irányított garázskapu adott modelljével fogunk dolgozni. A rendszer két fő részből áll, az egyik maga az automata, a másik pedig az operátor, azaz az autóvezető. Miután az autós beáll a garázsba a csukott sorompó elé, meg kell nyomnia egy gombot a jegykiadó gépen, amit az automata érzékel, és egy tálcára kiad egy parkolójegyet. A jegy elvételének hatására a sorompó felnyílik, az autós behajthat és leparkolhat. A sorompó lecsukódik, és az automata várja a következő leparkolni kívánt járművet.
A folyamatot leíró modellt meg fogjuk vizsgálni normál körülmények között, azaz amikor hibamentes esetet ír le a modellünk, illetve úgy is, amikor hibák vannak beiktatva a működésbe.

2. Hibamentes működés

Hibamentes működésről akkor beszélünk, amikor az egész folyamat pontosan úgy viselkedik, ahogy azt előre megjósolnánk. Nem lép fel hibás működés sem az automata részéről, sem pedig az operátor felöli oldalról.
A feladatunk, hogy feldolgozzuk és értelmezzük a rendszer Petri-hálóját, és elkészítsünk a modellből egy olyan eseménynaplót, amit a ProM keretrendszerbe betöltve, az ott lévő pluginek a log alapján egy a kiindulási hálóhoz képest, működésben ugyanolyan Petri-hálót készítenek el. Vizsgáljuk meg a hibamentes esetre vonatkozó modellt, amit az 1-es ábra illusztrál.



1. ábra – Garázskapu rendszer hibamentes modellje

A sárga részben találjuk meg az autósra vonatkozó leírást, míg a zöldben az automatához tartozó akciókat. A folyamat maga lineáris, az ábrán látható párhuzamos elrendezés csupán az átláthatóság kedvéért lett így megoldva.
Tranzíciók:
t_auto_beall: Ennek az átmenetnek az előfeltétele, hogy a „P_auto_erkezik" helyen legyen token (azaz van autó, amely parkolni szeretne), valamint az automata oldalán a „P_autovar" helyén is kell lennie egy tokennek, ami azt jelenti, hogy az automata képes autót fogadni.
t_gombnyomasra_var: Ahhoz, hogy engedélyezve legyen ez az átmenet, a „P_autovar" és a „P_autobe" helyeken kell lenniük tokeneknek. Ekkor az automata a gombnyomásra váró állapotba kerül.
t_gombnyomas: Előfeltétele, hogy a „P_gomb_vilagit" helyen legyen token, és ez teljesülni is fog esetünkben, hiszen a „t_auto_beall" következményeként ide, és az automata oldalán a „P_autobe" helyekre került egy-egy token. Ennek az átmenetnek a következménye pedig, hogy token fog kerülni a „P_jegyre_var" és a „P_gombbe" helyekre. Az autós tehát itt látja, hogy a gomb világít így megnyomja, az automata pedig érzékeli ezt.
t_jegykiadas: Az automata az előfeltételek teljesítése után kinyomtat egy parkoló cédulát, amit egy tálcára helyez el. Ennek hatására az automata a „P_elveszvar" állapotba kerül, ami az egyik előfeltétele a következő átmenetnek, a „t_jegyelvétel"-nek.
t_jegyelvétel: Az autós itt elveszi a kinyomtatott parkolójegyet. Tudjuk, hogy ennek az átmenetnek az előfeltétele, hogy legyen token az előbb említett „P_elveszvar" helyen, valamint a „P_jegyre_var" helyen, az autós oldalán.
t_sorompo_fel: Az automata a „P_jegyelvesz" és a „P_elveszvar" helyek hatására felnyitja a sorompót, majd a vezérjel a „P_beenged" helyre kerül.
t_behajtas: Az autós ekkor hajt át a már nyitott sorompó alatt, amely most már biztosan nyitva lesz, hiszen előfeltételei a „P_beenged" az automata oldalán, és a „P_sorompo_felnyitva" az operátor oldalán egyaránt teljesültek. A „t_behajtas" következményeként egy-egy token kerül a „P_auto_garazsban" és „P_auto_behajtva" helyeken. Előbbi az operátor, utóbbi az automata részéről.
t_sorompo_le: Az automata most már lecsukhatja a sorompót, hiszen előfeltételei teljesültek: A „P_beenged" és a „P_auto_behajtva" helyeken is volt token. Miután leengedte a sorompót, az automata visszatér a „P_auto_var" állapotba.
t_parkol: Az autós a garázsban keres egy szabad parkolóhelyet és leparkol. Ezután a „P_auto_leparkolva" helyre kerül a token, ami annyi vezérjelet tárol, ahány gépjármű már leparkolt a garázsban.

2.1. Hibamentes működés eseménynaplója

Az 1. lépés a résztvevő tagok meghatározása. Ezek a tagok fognak majd bekerülni az <Originator></Originator> MXML tagok közé. Ebben az esetben nekünk két féle résztvevő tagunk lesz, mégpedig az „Auto" (azaz az operátor, vagyis az autóvezetői oldal) és „Automata" (azaz a jegykiadó automata felöli oldal).
Rendszerezzük, hogy az egyes résztvevő tagokhoz milyen tranzíciók fognak tartozni. (A „t_" előtagot a nevekről, ami az átmeneteket jelöli, innentől elhagyjuk.)

Auto
auto_beall
gombnyomas
jegyelvetel
behajtas
parkol
Automata
gombnyomasra_var
jegykiadas
sorompo_fel
sorompo_le

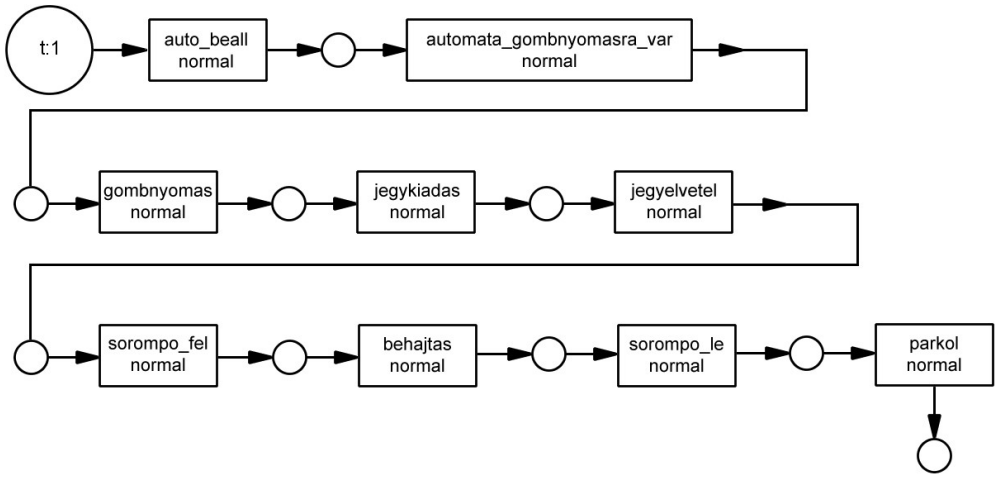
Ezek az események fognak bekerülni a naplófájlban a <WorkflowModelElement> </WorkflowModelElement> MXML tagok közé. A kész log-unk egyetlen autós érkezését fogja majd szimulálni, ezért a folyamati esetből (ProcessInstance) is csak egyetlen példányra van szükségünk. Az események típusát (<EventType>) meghatározhatjuk az egyes MXML bejegyzések (<AuditTrailEntry>) felépítésekor is, de már most is látszódik, hogy minden egyes eseményünknek megfelelő a „normal" attribútum is, hiszen egyetlen összetettebb esemény sincs a rendszerben, amit pl. start - complete eseménytípusokkal kellene felbontani.
Mivel az eredeti Petri-háló szimulálása során az egyes tüzelések egy előző tüzelés eredményeként fejeződnek be, ezért az MXML kódból generált Petri-háló nyilvánvalóan lineáris lesz. Az MXML generátor segédprogrammal elkészíthetjük a log fájlt.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- MXML version 1.0 -->
<!-- This is a process enactment event log created to be analyzed by ProM. -->
<WorkflowLog xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://is.tm.tue.nl/research/processmining/WorkflowLog.xsd"
description="Created by manual, using an existing Petri net">
<Source program="MXML generator" />
<Process id="DEFAULT" description="Simulated process">
<ProcessInstance id="1" description="Simulated process instance">
<AuditTrailEntry>
  <WorkflowModelElement>auto_beall</WorkflowModelElement>
  <EventType>normal</EventType>
  <TimeStamp>2010-02-10T13:46:25.000+01:00</TimeStamp>
  <Originator>Auto</Originator>
</AuditTrailEntry>
<AuditTrailEntry>
  <WorkflowModelElement>automata_gombnyomasra_var
  </WorkflowModelElement>
  <EventType>normal</EventType>
  <TimeStamp>2010-02-10T13:47:00.000+01:00</TimeStamp>
  <Originator>Automata</Originator>
</AuditTrailEntry>
<AuditTrailEntry>
  <WorkflowModelElement>gombnyomas</WorkflowModelElement>
  <EventType>normal</EventType>
  <TimeStamp>2010-02-10T13:47:10.000+01:00</TimeStamp>
  <Originator>Auto</Originator>
</AuditTrailEntry>
<AuditTrailEntry>
  <WorkflowModelElement>jegykiadas</WorkflowModelElement>
  <EventType>normal</EventType>
  <TimeStamp>2010-02-10T13:47:15.000+01:00</TimeStamp>
  <Originator>Automata</Originator>
</AuditTrailEntry>
<AuditTrailEntry>
  <WorkflowModelElement>jegyelvetel</WorkflowModelElement>
  <EventType>normal</EventType>
  <TimeStamp>2010-02-10T13:47:20.000+01:00</TimeStamp>
  <Originator>Auto</Originator>
</AuditTrailEntry>
<AuditTrailEntry>
  <WorkflowModelElement>sorompo_fel</WorkflowModelElement>
  <EventType>normal</EventType>
  <TimeStamp>2010-02-10T13:47:30.000+01:00</TimeStamp>
```

```
  <Originator>Automata</Originator>
</AuditTrailEntry>
<AuditTrailEntry>
  <WorkflowModelElement>behajtas</WorkflowModelElement>
  <EventType>normal</EventType>
  <TimeStamp>2010-02-10T13:47:55.000+01:00</TimeStamp>
  <Originator>Auto</Originator>
</AuditTrailEntry>
<AuditTrailEntry>
  <WorkflowModelElement>sorompo_le</WorkflowModelElement>
  <EventType>normal</EventType>
  <TimeStamp>2010-02-10T13:48:25.000+01:00</TimeStamp>
  <Originator>Automata</Originator>
</AuditTrailEntry>
<AuditTrailEntry>
  <WorkflowModelElement>parkolas</WorkflowModelElement>
  <EventType>normal</EventType>
  <TimeStamp>2010-02-10T13:50:35.000+01:00</TimeStamp>
  <Originator>Auto</Originator>
</AuditTrailEntry>
</ProcessInstance>
</Process>
</WorkflowLog>
```

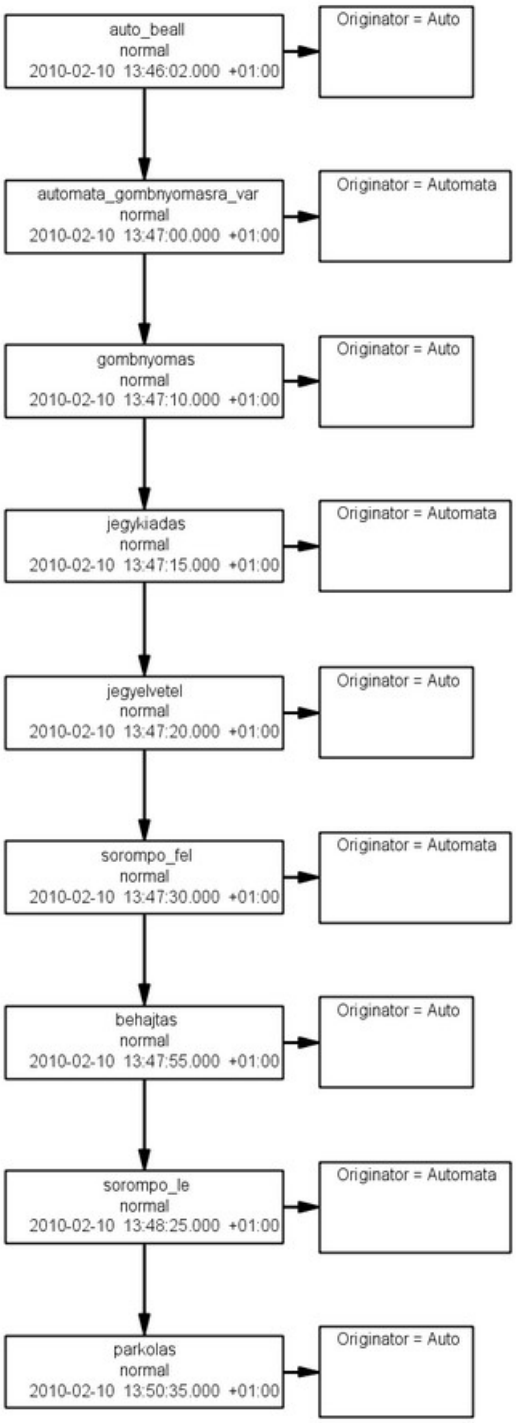
2. ábra – Hibamentes működés eseménynaplója

Ezután be tudjuk tölteni a logot a ProM keretrendszerbe és ellenőrizzük, hogy az ott lévő α-algoritmus kimeneti modellje megegyezik-e a kiindulási modellünkkel. Miután ezt megtettük a következő ábrát fogjuk kapni eredményül, amit a 3-as ábra szemléltet.



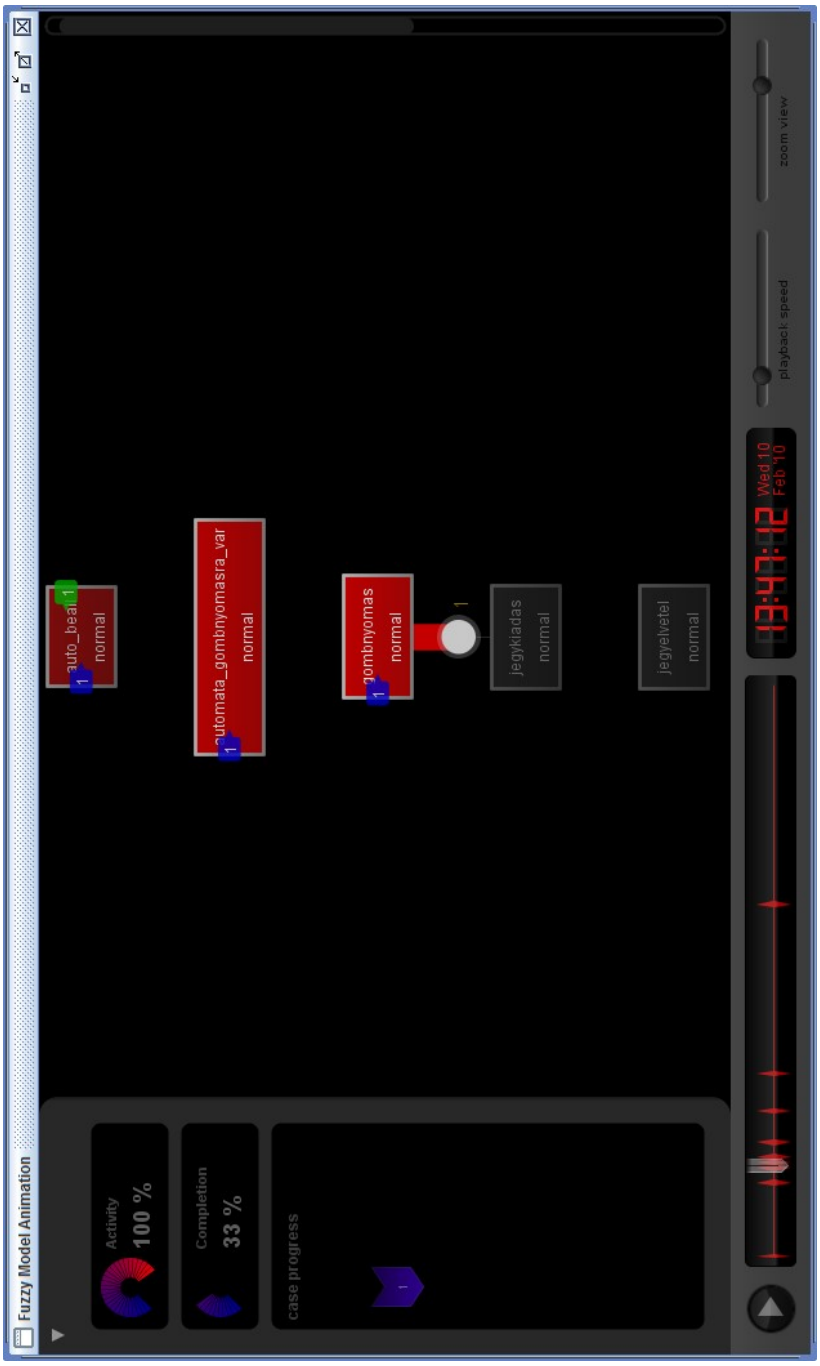
3. ábra – Hibamentes működés, ProM α-algoritmusának kimeneti Petri-hálója

A modellt szemügyre véve láthatjuk, hogy az eredmény lineáris, a tranzakciók egymást követik a folyamat lefutásában, tehát a naplófájlunk jónak ígérkezik.
A ProM **„Analysis”** menüpontjában lévő **„Open log with classic dialog”** menüpontra kattintva egy hasonló ábrát kaphatunk, csak abban minden részletet láthatunk, amit a log-ba beírtunk. (Minden <AuditTrailEntry> tagokat.) A 4-es ábrán láthatók az eredmények.



4. ábra – Hibamentes működés, Analysis plugin

Ahhoz, hogy a vezérjelek mozgását is szimulálni tudjuk a generált hálóban, le kell futtatnunk a „Fuzzy Miner” algoritmust. Ha ezt megtettük a következő animációs modellt kapjuk, amit a 5-ös ábrán láthatunk.



5. ábra – Hibamentes működés, Fuzzy Miner animációs modellje (részlet)

2.2. Hibamentes működés – a log kibányászása

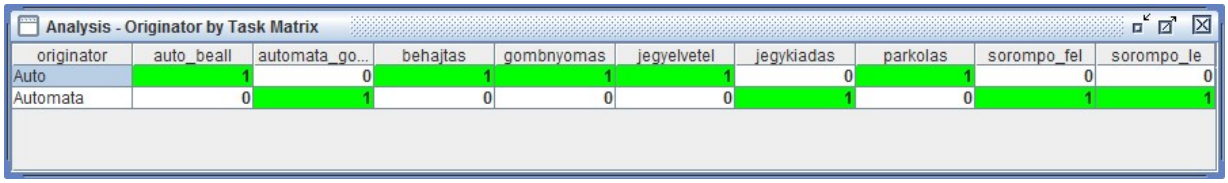
Megvizsgálhatjuk a ProM keretrendszer beépített plugin-jei segítségével, hogy mit is árul el nekünk az elkészült naplófájl.

Egy meglehetősen átfogó lényegi összegzést kaphatunk, ha kiválasztjuk a menüből az **„Analysis”** menüponot, majd azon belül az **„Open log with classic dialog”** további opciót. A következőket olvashatjuk le az összegző ablakról:

- Folyamatok száma: 1
- AuditTrail bejegyzések száma: 9 (vagyis az átmenetek száma)
- Fájlnév: log_hibamentes.mxml
- Leírás: Létező Petri-háló alapján, manuális úton készült (ezt mi írtuk bele a log-ba)
- Forrás: MXML Generátor (szintén a log egyik attribútumából olvassa ki)

- Láthatjuk felsorolva a log bejegyzések adatait, azaz az átmeneteket, az események típusát és előfordulásukra vonatkozó adatokat
- A kezdő és befejező eseményeket (auto_beall és parkolas)
- A résztvevő tagok (Originators) számát, ami ugye kettő (Auto és Automata)
- Láthatjuk, hogy az „Auto”-hoz öt, míg az „Automata”-hoz négy esemény tartozik.

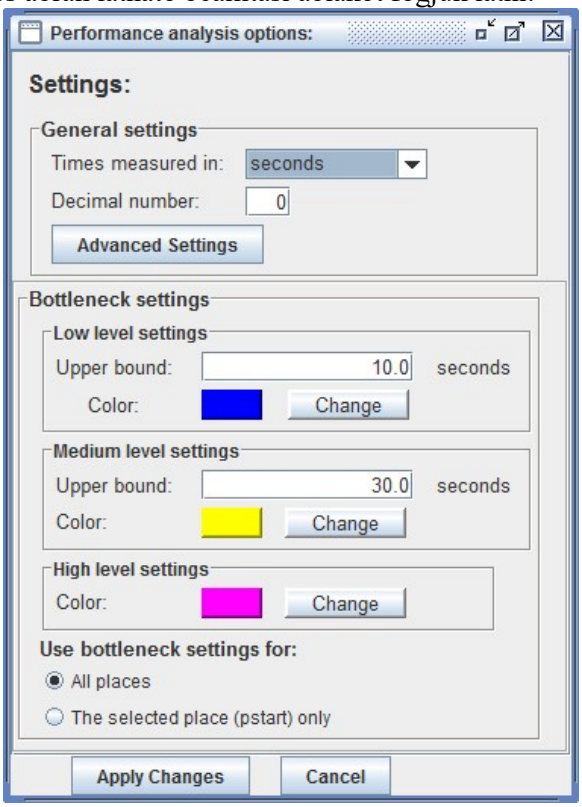
Hogyan nézhetjük meg azt, hogy az „Auto”-hoz, vagy az „Automata”-hoz pontosan mely események tartoznak? Nos, erre is megvan a megfelelő plugin, amit
Az **„Analysis”** menüpontból az **„Originator by task Matrix”** menüre kattintva megnézhetjük, hogy az Auto-hoz vagy az Automata-hoz pontosan mely események tartoznak (6-os ábra).



originator	auto_beall	automata_go...	behajtas	gombnyomas	jegyelvetel	jegykiadas	parkolas	sorompo_fel	sorompo_le
Auto	1	0	1	1	1	0	1	0	0
Automata	0	1	0	0	0	1	0	1	1

6. ábra – Hibamentes működés, Originator by Task Matrix plugin

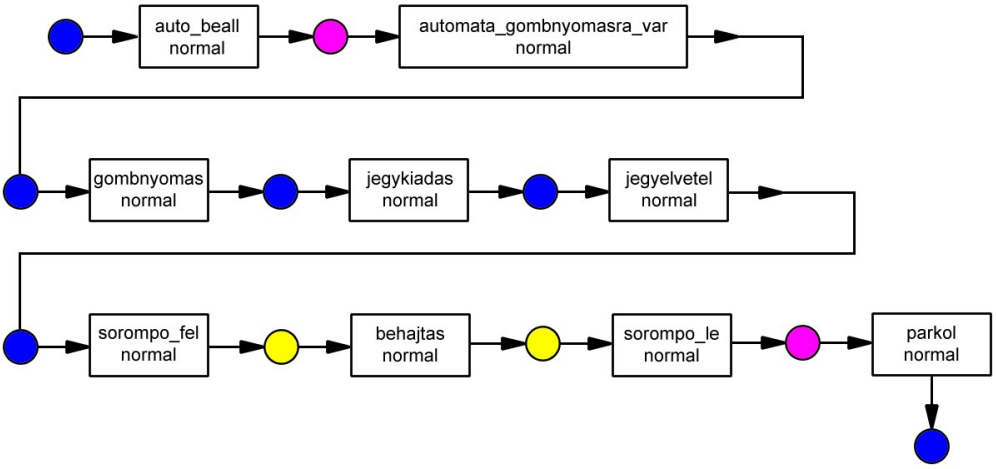
A következő plugin abban segít, hogy felderítsük az időtorlódásra vonatkozó elemzéseket. Ehhez szükségünk van egy referencia modellre, amit az α-algoritmus kimeneti Petri-hálója fog biztosítani számunkra. Tehát első lépésként futtassuk le a log-ra ezt az algoritmust, vagyis **„Mining”** menüpont, majd **„Alpha Algorithm Plugin”** opció. Miután ezzel megvagyunk és kész a modellünk, valamint meggyőződtünk arról, hogy a modellt tartalmazó ablakunk az aktív, kattintsunk az **„Analysis”** menüre, majd **„Selected Petri net”**, ezután pedig **„Performance Analysis with Petri Net”**. Nyomjuk meg az ablak bal alsó sarkában lévő **„Start analysis”** gombot. Ekkor az algoritmus az alapértelmezett adatokkal elkészíti a modellt. A **„Settings”** gombra kattintva elvégezhetünk bizonyos beállításokat. Miután megnyomtuk, a 7-es ábrán látható beállítási ablakot fogjuk látni.



7. ábra – Hibamentes működés, Teljesítményelemzés plugin, beállítások

Az első részben két dolgot állíthatunk be, az egyik az időegységre vonatkozó mértékegység, a másik pedig a tizedes jegyek száma. Mivel az eseménynaplóba mi magunk írtuk bele az időbélyegeket (hiszen a kiindulási modellünk nem tartalmazott időegységekre vonatkozó leírásokat), tudjuk, hogy az

egész folyamat nem tart tovább pár percnél, így érdemes másodpercre állítani. A tizedes jegyek számát állítsuk nullára, hiszen most lényegi szerepe nincsen.
A következő rész a torlódásra vonatokozó beállítási lehetőségek. Itt háromféle küszöbértéket adhatunk meg, amelyekkel szabályozhatjuk, hogy az egyes események időbeli lefutása milyen szintnek (alacsony, közepes, magas) feleljen meg. Vagyis, ha az alacsony szinthez tartozó kék színnel jelölt értékhez 10 másodpercet állítunk be, akkor a modellen azon helyek színe, ahol egy esemény maximum 10 másodpercig tartott, kék színű lesz. Hasonlóan működik a közepes szinthez tartozó sárga színű érték is, ide állítsunk 30 másodpercet. A magenta színű érték, mely a magas szintet képviseli, automatikusan 30 másodpercnél nagyobb értékekre fog vonatkozni.
A legutolsó résznél pedig megadhatjuk, hogy beállításaink minden eseményre, vagy csak egy általunk kiválasztottra legyen érvényes. Ha elvégeztünk minden beállítást, kattintsunk az ablak alján található **„Apply changes”** gombra. Ezzel készen is van az immáron színeket is tartalmazó Petri-hálónk. Ezt láthatjuk a 8-as ábrán.



8. ábra – Hibamentes működés, Teljesítményelemzés plugin

A színek segítségével a modellről azonnal leolvashatjuk, hogy mely események fognak a leghosszabb ideig (magenták) tartani, melyek a 10 és 30 másodperc közöttiek (sárgák), illetve melyek 10, vagy ennél kevesebb idő alattiak (kékek). Láthatjuk például, hogy míg az autós eljut az automatáig, ami érzékeli őt és gombnyomásra aktívvá teszi a jegykiadó gombot, több mint 30 másodperc telik el. Hogy pontosan mennyi, megtudhatjuk azáltal, hogy rákattintunk erre a helyre, és az ablak alsó felében lévő táblázatból pontosan leolvashatjuk. (Ha az eseménynaplónk több autós parkolását is tartalmazná, ahol természetesen különbözőek lennének az időbélyegek, a táblázatból leolvashatjuk a minimum, maximum és átlag időegységre vonatkozó értékeket.) Jelen esetben ez 35 másodperc, amit torlódáspontként is felfoghatunk. A hatékonyabb működéshez tehát a való életben itt valamilyen szintű gyorsításra lenne szükség.
Megfigyelhetjük, hogy a sorompó felnyitása után az autósnak 25 másodpercébe telik, míg behajt a garázsba. Ez sárga színű, tehát közepes szintnek felel meg, ám ez egyébként sem a rendszer jellemzője, hanem az autóvezetőé, csakúgy, mint a sorompó lecsukása utáni parkolás esemény ideje (ami már 130 másodperc).
Ezzel az algoritmussal tehát hatékonyan és pontosan megvizsgálhatjuk, hogy hol lehetnek a folyamatban torlódáspontok.

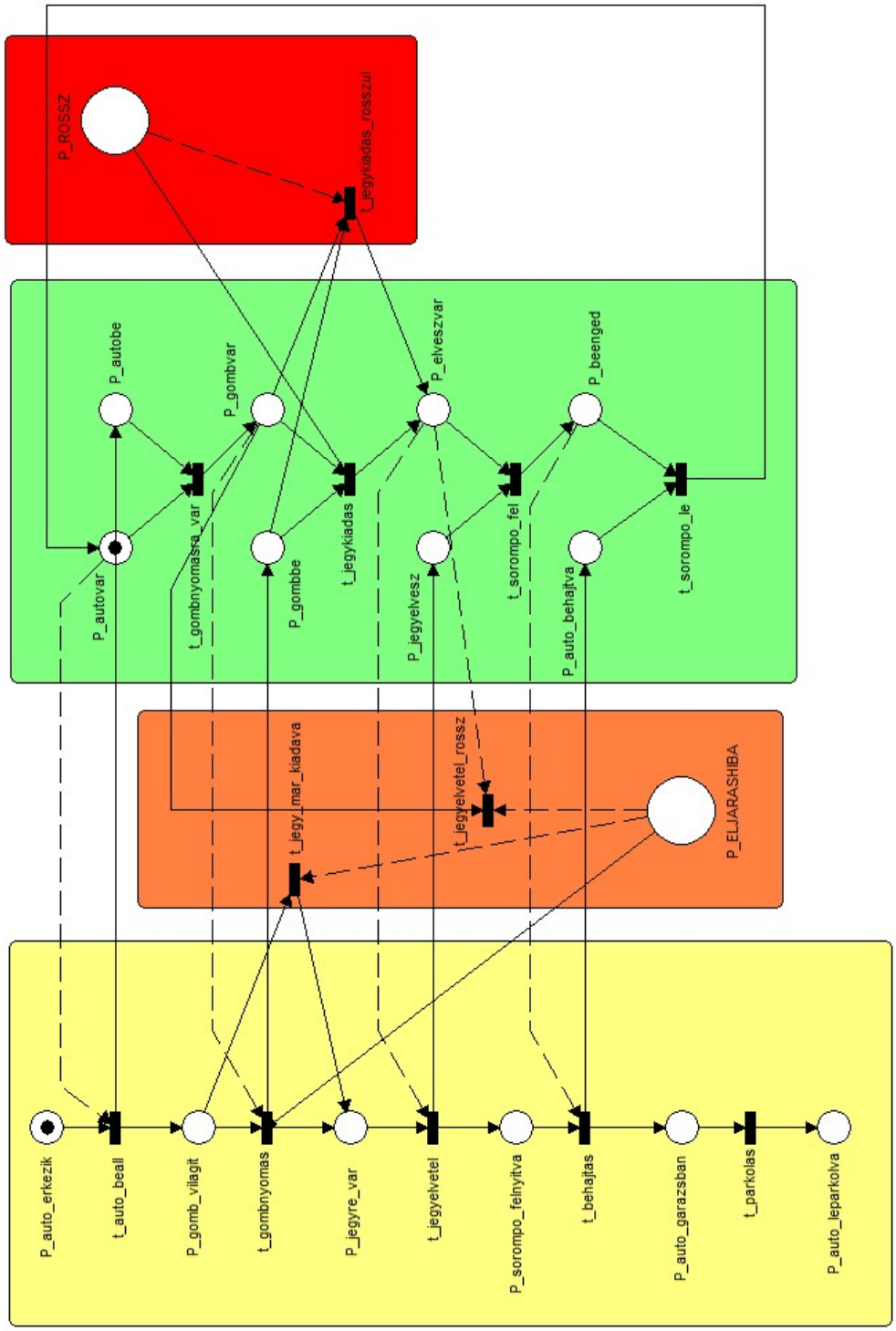
2.3. Események egyenkénti elhagyása

Megvizsgálhatjuk a folyamatot az szerint is, hogy ha egyesével elhagyjuk az egyes eseményeket, az vajon milyen változást fog előidézni a folyamat lefutásának szempontjából, valamint milyen kihatással lesz magára a folyamatbányászatra.
„Auto_beall” elhagyása: A rendszerben a legfontosabb átmenet, hiszen a parkoló autóst szimbolizálja, nélküle nem történhetne meg maga a parkolási folyamat. Az automata sem vált át gombnyomásra váró állapotban, hiszen nem érzékel autóst.

„Automata_gombnyomasra_var” elhagyása: Ha ezt az átmenet megszüntetnénk, akkor az érkező autós bármennyit várhat, hogy a gomb aktív legyen, sohasem lesz az, mivel az automata nem fogja érzékelni őt. Tehát a folyamat meg fog állni, még a gombnyomás előtt.
„Gombnyomas” elhagyása: Gombnyomás nélkül bizony sohasem fog jegyet kapni az autós, így ez az állapot is kritikusnak számít működés szempontjából.
„Jegykiadas” elhagyása: Ugyancsak a folyamat megállását eredményezi, hiszen ez az állapot is feltétele a sorompó felnyitásának.
„Jegyelvetel” elhagyása: Mivel a sorompó úgy működik, hogy érzékeli a jegy elvételét a tálcáról, majd csak ezután nyit fel, a folyamat itt is megakad.
„Sorompo_fel” elhagyása: Magától értetődően, a sorompó sohasem fog felnyílni, így az autós megreked a jegyelvétel után.
„Behajtas” elhagyása: Ez az esemény elhagyása azzal jár, hogy az autó megáll a sorompó előtt, tehát a maga a folyamat is leáll.
„Sorompo_le” elhagyása: Az egyetlen olyan átmenet, amit ha kiiktatnánk, az autós le tudna parkolni, hiszen miután elvette a jegyet és felnyílt a sorompó, behajt és leparkol. A lecsukással már nem befolyásolja az aktuális folymatot.
„Parkol” elhagyása: Maga a parkolás átmenet elhagyását csak úgy lehetne értelmezni, ha az autós a behajtás után, rögtön el is hagyja a garázst.
Bármelyik eseményt is hagyjuk el, a beépített algoritmusok ugyan ki fognak rajzolni valamilyen modellt, és a „Fuzzy Miner” plugin is lefuttatja az animációt, azonban az elemzés során be fogjuk látni, hogy a folyamat így nem értelmezhető. Mivel megállapítottuk, hogy ez egy lineáris lefolyású eset, ahol az egyes akciók előfeltételei az azokat megelőző akciók, így a modellben egyik eseményt sem lehet elhagyni vagy mással helyettesíteni.

3. Hibás működés

A parkoló rendszer hibás működése annyiban fog eltérni az alap modelltől, hogy kétféle hibaeseményt fog tartalmazni. Ezt úgy fogjuk megvalósítani az eseménynaplóban, hogy három különböző folyamati esetet (ProcessInstance) fogunk létrehozni. Az első részfolyamat a hibamentes esetet fogja tartalmazni, a második és harmadik részfolyamat pedig a hibás eseteket, amelyek mint majd látni fogjuk egymással kapcsolatban állnak. A 9-es ábra a kiindulási modellt ábrázolja.
Az utolsó doboz, azaz a piros rész fogja azt a hibát észlelni, amikor a vezető megnyomja a gombot, ám az automata egy jegy helyett kettőt fog kinyomtatni. Ekkor az autós elveszi az egyiket, majd a folyamat folytatódik tovább. Egy jegy azonban ott marad a „P_elveszvar” helyen, és ez azt fogja jelenteni, hogy a következő autós számára már eleve ott lesz egy parkolójegy a tálcán. Persze megnyomhatja újra a gombot, ebben az esetben a kiadott jegyek száma mindig növekedni fog. Egy új átmenet lesz, a „t_jegykiadas_rosszul.”
A modellben a narancssárga rész (második doboz) fogja reprezentálni azt a hibát, amikor az egyik lépést teljesen kiiktatjuk („t_gombnyomas”). Ez azt jelenti, hogy az autós előbb veszi el a jegyet, mielőtt még megnyomná a gombot. Természetesen ez csak abban az esetben lehetséges, ha az automata már eleve hibásan működött (piros hiba). Ekkor az autós, az aktívvá vált gomb után annak megnyomása nélkül, az előzőleg kiadott jegyet veszi el. Ez a hiba csak addig állhat fenn, amíg a „P_elveszvar” helyen el nem fogy a token (azaz el nem fogynak a már hibásan kiadott jegyek). Két új átmenet fog megjelenni a modellben, a „t_jegy_mar_kiadva” illetve a „t_jegyelvetel_rossz”.
Mivel az előzőkben az egyszerűség kedvéért egyébként is csak egyszeri lefutásra vizsgáltuk meg a folyamatot, azaz a hibamentes működés esetén is csak egy autós parkolását szimuláltuk, ebben az esetben is úgy fogjuk elkészíteni az eseménynaplónkat, hogy első esetben lefut hibamentesen a folyamat, majd második esetben ott marad egy jegy a dupla jegykiadás miatt. A harmadik esetben a dupla jegykiadás miatt ottmaradt jegyet fogjuk elvenni gombnyomás helyett.



9. ábra – Garázskapu rendszer hibákat észlelő modellje

3.1. Hibás működés eseménynaplója, bányászata

A három folyamat kapcsolódik egymáshoz, azaz az első részfolyamat a hibamentes esetet fogja leírni, a második részfolyamat tartalmazza azt a hibát, amikor az automata két jegyet ad ki, a harmadik, pedig amikor ezt a plusz jegyet vesszük el gombnyomás nélkül.

<ProcessInstance id="1" description="Hibamentes mukodes">

Hibamentes: *auto_beall → automata_gombnyomasra_var → gombnyomas → jegykiadas → jegyelvetel → sorompo_fel → behajtas → sorompo_le → parkolas*

<ProcessInstance id="2" description="Dupla jegykiadas">

Első hiba (piros): *auto_beall → automata_gombnyomasra_var → gombnyomas → jegykiadas_rosszul → jegyelvetel → sorompo_fel → behajtas → sorompo_le → parkolas*

<ProcessInstance id="3" description="Jegy mar kiadva">
Második hiba (narancs): *auto_beall → automata_gombnyomasra_var → jegy_mar_kiadva → jegyelvetel_rossz → sorompo_fel → behajtas → sorompo_le → parkolas*
==========

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- MXML version 1.0 -->
<!-- This is a process enactment event log created to be analyzed by ProM. -->
<WorkflowLog xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://is.tm.tue.nl/research/processmining/WorkflowLog.xsd"
description="Created by manual, using an existing Petri net">
<Source program="MXML generator" />
<Process id="DEFAULT" description="Simulated process">

<ProcessInstance id="1" description="Hibamentes mukodes">
<AuditTrailEntry>
  <WorkflowModelElement>auto_beall</WorkflowModelElement>
  <EventType>normal</EventType>
  <TimeStamp>2010-02-10T13:46:25.000+01:00</TimeStamp>
  <Originator>Auto</Originator>
</AuditTrailEntry>
<AuditTrailEntry>
  <Data><Attribute name = "jegyekSzama">1 </Attribute>
  </Data>
<WorkflowModelElement>automata_gombnyomasra_var
</WorkflowModelElement>
  <EventType>normal</EventType>
  <TimeStamp>2010-02-10T13:47:00.000+01:00</TimeStamp>
  <Originator>Automata</Originator>
</AuditTrailEntry>
<AuditTrailEntry>
  <WorkflowModelElement>gombnyomas</WorkflowModelElement>
  <EventType>normal</EventType>
  <TimeStamp>2010-02-10T13:47:10.000+01:00</TimeStamp>
  <Originator>Auto</Originator>
</AuditTrailEntry>
<AuditTrailEntry>
  <WorkflowModelElement>jegykiadas</WorkflowModelElement>
  <EventType>normal</EventType>
  <TimeStamp>2010-02-10T13:47:15.000+01:00</TimeStamp>
  <Originator>Automata</Originator>
</AuditTrailEntry>
<AuditTrailEntry>
  <WorkflowModelElement>jegyelvetel</WorkflowModelElement>
  <Data><Attribute name = "jegyekSzama">0 </Attribute>
  </Data>
  <EventType>normal</EventType>
  <TimeStamp>2010-02-10T13:47:20.000+01:00</TimeStamp>
  <Originator>Auto</Originator>
</AuditTrailEntry>
<AuditTrailEntry>
  <WorkflowModelElement>sorompo_fel</WorkflowModelElement>
  <EventType>normal</EventType>
  <TimeStamp>2010-02-10T13:47:30.000+01:00</TimeStamp>
  <Originator>Automata</Originator>
</AuditTrailEntry>
```

```
<AuditTrailEntry>
  <WorkflowModelElement>behajtas</WorkflowModelElement>
  <EventType>normal</EventType>
  <TimeStamp>2010-02-10T13:47:55.000+01:00</TimeStamp>
  <Originator>Auto</Originator>
</AuditTrailEntry>
<AuditTrailEntry>
  <WorkflowModelElement>sorompo_le</WorkflowModelElement>
  <EventType>normal</EventType>
  <TimeStamp>2010-02-10T13:48:25.000+01:00</TimeStamp>
  <Originator>Automata</Originator>
</AuditTrailEntry>
<AuditTrailEntry>
  <WorkflowModelElement>parkolas</WorkflowModelElement>
  <EventType>normal</EventType>
  <TimeStamp>2010-02-10T13:50:35.000+01:00</TimeStamp>
  <Originator>Auto</Originator>
</AuditTrailEntry>
</ProcessInstance>

<ProcessInstance id="2" description="Dupla jegykiadas">
<AuditTrailEntry>
  <WorkflowModelElement>auto_beall</WorkflowModelElement>
  <EventType>normal</EventType>
  <TimeStamp>2010-02-10T14:46:02.000+01:00</TimeStamp>
  <Originator>Auto</Originator>
</AuditTrailEntry>
<AuditTrailEntry>
<WorkflowModelElement>automata_gombnyomasra_var
</WorkflowModelElement>
  <EventType>normal</EventType>
  <TimeStamp>2010-02-10T14:47:00.000+01:00</TimeStamp>
  <Originator>Automata</Originator>
</AuditTrailEntry>
<AuditTrailEntry>
  <WorkflowModelElement>gombnyomas</WorkflowModelElement>
  <EventType>normal</EventType>
  <TimeStamp>2010-02-10T14:47:10.000+01:00</TimeStamp>
  <Originator>Auto</Originator>
</AuditTrailEntry>
<AuditTrailEntry>
  <Data><Attribute name = "jegyekSzama">2 </Attribute>
  </Data>
  <WorkflowModelElement>jegykiadas_rosszul</WorkflowModelElement>
  <EventType>normal</EventType>
  <TimeStamp>2010-02-10T14:47:15.000+01:00</TimeStamp>
  <Originator>Automata</Originator>
</AuditTrailEntry>
<AuditTrailEntry>
  <Data><Attribute name = "jegyekSzama">1 </Attribute>
  </Data>
  <WorkflowModelElement>jegyelvetel</WorkflowModelElement>
  <EventType>normal</EventType>
  <TimeStamp>2010-02-10T14:47:20.000+01:00</TimeStamp>
  <Originator>Auto</Originator>
```

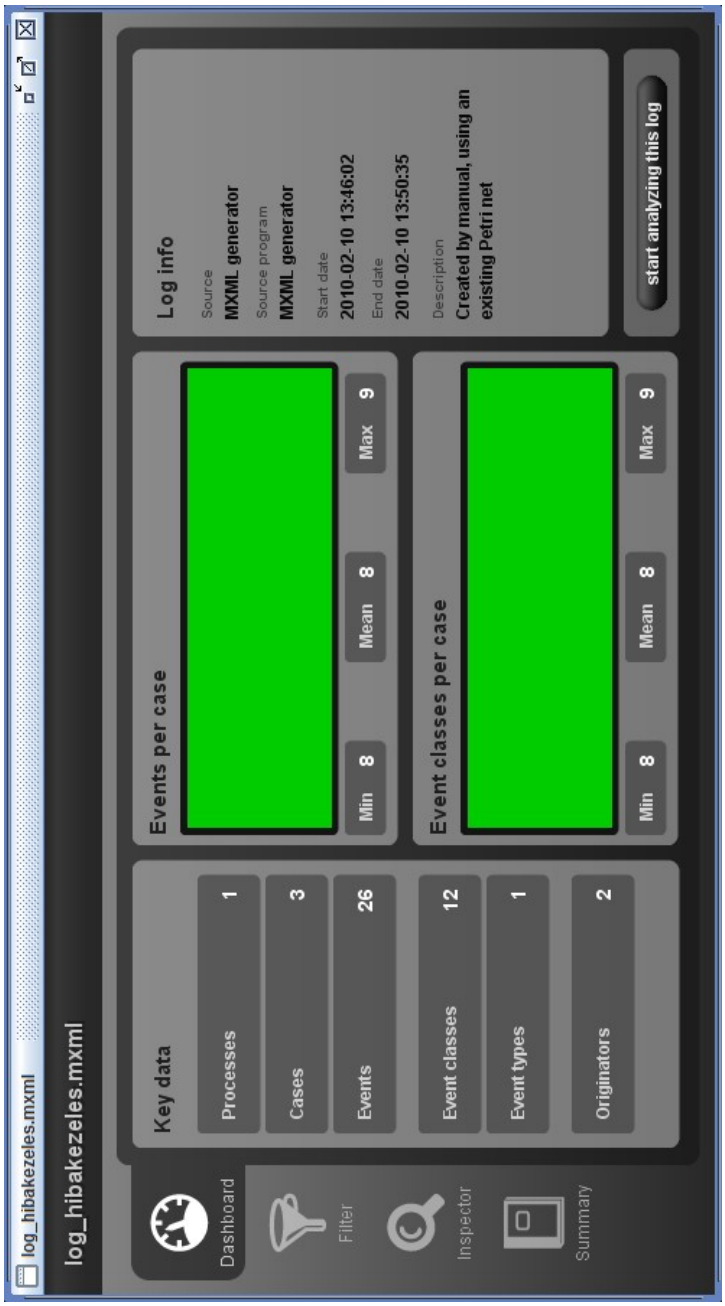
```
</AuditTrailEntry>
<AuditTrailEntry>
  <WorkflowModelElement>sorompo_fel</WorkflowModelElement>
  <EventType>normal</EventType>
  <TimeStamp>2010-02-10T14:47:30.000+01:00</TimeStamp>
  <Originator>Automata</Originator>
</AuditTrailEntry>
<AuditTrailEntry>
  <WorkflowModelElement>behajtas</WorkflowModelElement>
  <EventType>normal</EventType>
  <TimeStamp>2010-02-10T14:47:55.000+01:00</TimeStamp>
  <Originator>Auto</Originator>
</AuditTrailEntry>
<AuditTrailEntry>
  <WorkflowModelElement>sorompo_le</WorkflowModelElement>
  <EventType>normal</EventType>
  <TimeStamp>2010-02-10T14:48:25.000+01:00</TimeStamp>
  <Originator>Automata</Originator>
</AuditTrailEntry>
<AuditTrailEntry>
  <WorkflowModelElement>parkolas</WorkflowModelElement>
  <EventType>normal</EventType>
  <TimeStamp>2010-02-10T14:50:35.000+01:00</TimeStamp>
  <Originator>Auto</Originator>
</AuditTrailEntry>
</ProcessInstance>

<ProcessInstance id="3" description="Jegy mar kiadva">
<AuditTrailEntry>
  <WorkflowModelElement>auto_beall</WorkflowModelElement>
  <EventType>normal</EventType>
  <TimeStamp>2010-02-10T15:46:02.000+01:00</TimeStamp>
  <Originator>Auto</Originator>
</AuditTrailEntry>
<AuditTrailEntry>
<WorkflowModelElement>automata_gombnyomasra_var
</WorkflowModelElement>
  <EventType>normal</EventType>
  <TimeStamp>2010-02-10T15:47:10.000+01:00</TimeStamp>
  <Originator>Auto</Originator>
</AuditTrailEntry>
<AuditTrailEntry>
  <Data><Attribute name = "jegyekSzama">1 </Attribute>
  </Data>
  <WorkflowModelElement>jegy_mar_kiadva</WorkflowModelElement>
  <EventType>normal</EventType>
  <TimeStamp>2010-02-10T15:47:15.000+01:00</TimeStamp>
  <Originator>Automata</Originator>
</AuditTrailEntry>
<AuditTrailEntry>
  <Data><Attribute name = "jegyekSzama">0 </Attribute>
  </Data>
  <WorkflowModelElement>jegyelvetel_rossz</WorkflowModelElement>
  <EventType>normal</EventType>
  <TimeStamp>2010-02-10T15:47:20.000+01:00</TimeStamp>
```

```
  <Originator>Auto</Originator>
</AuditTrailEntry>
<AuditTrailEntry>
  <WorkflowModelElement>sorompo_fel</WorkflowModelElement>
  <EventType>normal</EventType>
  <TimeStamp>2010-02-10T15:47:30.000+01:00</TimeStamp>
  <Originator>Automata</Originator>
</AuditTrailEntry>
<AuditTrailEntry>
  <WorkflowModelElement>behajtas</WorkflowModelElement>
  <EventType>normal</EventType>
  <TimeStamp>2010-02-10T15:47:55.000+01:00</TimeStamp>
  <Originator>Auto</Originator>
</AuditTrailEntry>
<AuditTrailEntry>
  <WorkflowModelElement>sorompo_le</WorkflowModelElement>
  <EventType>normal</EventType>
  <TimeStamp>2010-02-10T15:48:25.000+01:00</TimeStamp>
  <Originator>Automata</Originator>
</AuditTrailEntry>
<AuditTrailEntry>
  <WorkflowModelElement>parkolas</WorkflowModelElement>
  <EventType>normal</EventType>
  <TimeStamp>2010-02-10T15:50:35.000+01:00</TimeStamp>
  <Originator>Auto</Originator>
</AuditTrailEntry>
</ProcessInstance>
</Process>
</WorkflowLog>
```


10. ábra – Garázskapu rendszer hibákat észlelő eseménynaplója

Ezután betöltjük az eseménynaplót a ProM keretrendszerbe. Először az **„Analysis”** menüpont **„Open log with slicker dialog”** lehetőségét nézzük meg (11-es ábra).
Az ablak négy fő részből áll. Bal oldalt helyezkednek el a fő menüpontok. A 11. ábrán a kezdő, „Dashboard” felületet látjuk, majd alatta „Filter”, „Inspector” és „Summary” további analízist segítő pontok. A **„Dashboard”**-on láthatjuk, a legfontosabb összegző adatokat: 1 fő folyamat, 3 részfolyamat (Cases), illetve 26 esemény van összesen. 12 különböző átmenet (Event classes), 1 esemény típus (Event types), illetve 2 originátor (Originators). Középen láthatjuk, hogy a 3 részfolyamatba minimum 8 esemény tartozik, maximum 9, átlagban pedig szintén 8. Jobb oldalon a „Log info” résznél találjuk, hogy az eseménynapló a MXML generátor programmal készült (Source program), a folyamat időbeli kezdete és vége is fel van tüntetve (Start date, End date), illetve egy rövid leírás (Description).
A **„Filter”** menüpontban lehetőségünk van megszűrni a log-ot események, résztvevő tagok, esemény típusok szerint. Ez leginkább akkor érdekes, ha az eseménynapló túl sok információt tartalmaz, mint ami nekünk kellene a bányászathoz.



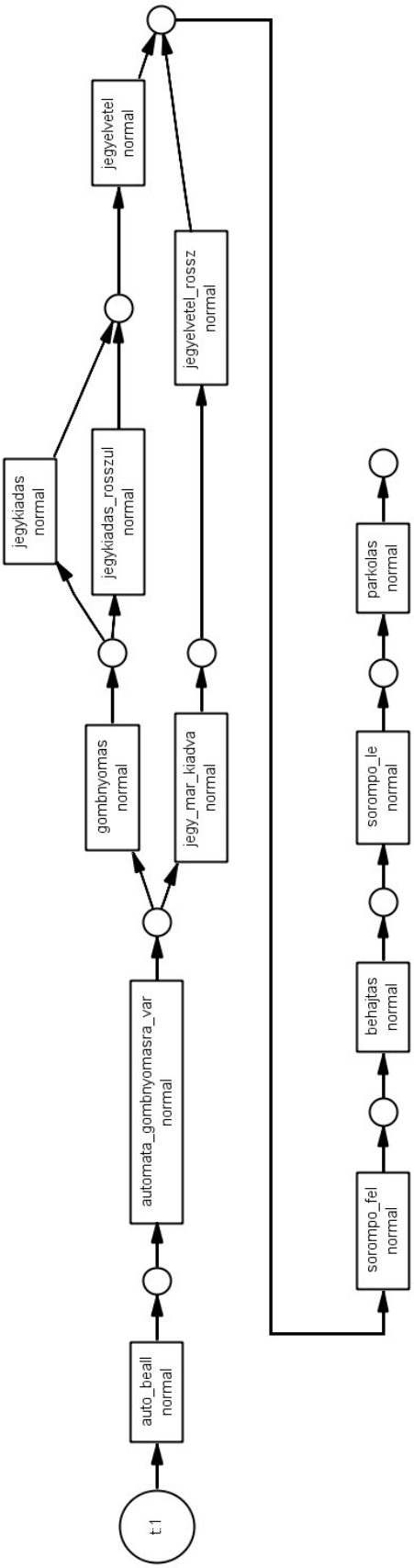
11. ábra – ProM analizáló felülete

Az **„Inspector”** menüpontban kicsit részletesebben is láthatjuk az eseménynapló elemzését. Itt már részfolyamatok szerint tekinthetjük meg az egyes eseményeket és ahhoz kapcsolódó adataikat. Ezen kívül a program kirajzolja nekünk egy színezett ábra segítségével, hogy mely tranzakciók milyen gyakorisággal fordulnak elő a logban. A zöld szín jelenti a gyakori eseményeket, a narancs a ritkábbakat. A 12-es ábrán láthatjuk ezt a diagramot, ahol mindhárom részfolyamatból kivan emeltve egy-egy esemény. Mindemellett egyértelműen jelezve van az ábrán, mégpedig az előtte lévő szürke részen, hogy melyik részábra melyik részfolyamathoz tartozik. Itt fel van tüntetve a folyamat azonosítószáma, illetve az események száma. Mint látható az első részfolyamatnál, azaz a hibamentes esetnél, a „sorompo_le” átmenet 100%-os gyakoriságú. A „gombnyomas” esemény, ami a második részfolyamatnál van kiemelve, már csak 66,67%-os gyakoriságú, hiszen tudjuk, hogy a harmadik részfolyamatban nem történik gombnyomás. Végezetül a „jegy_mar_kiadva” esemény azért csak 33,33%-os gyakoriságú, mert csakis a harmadik részfolyamatban található meg. Az aktív résznél egyébként az átmenetek nevein és százalékos gyakoriságán kívül láthatjuk meg az originátor értékét, illetve a hozzá tartozó dátumot, azaz időbélyeget is.

12. ábra – ProM analizáló felülete, Explorer

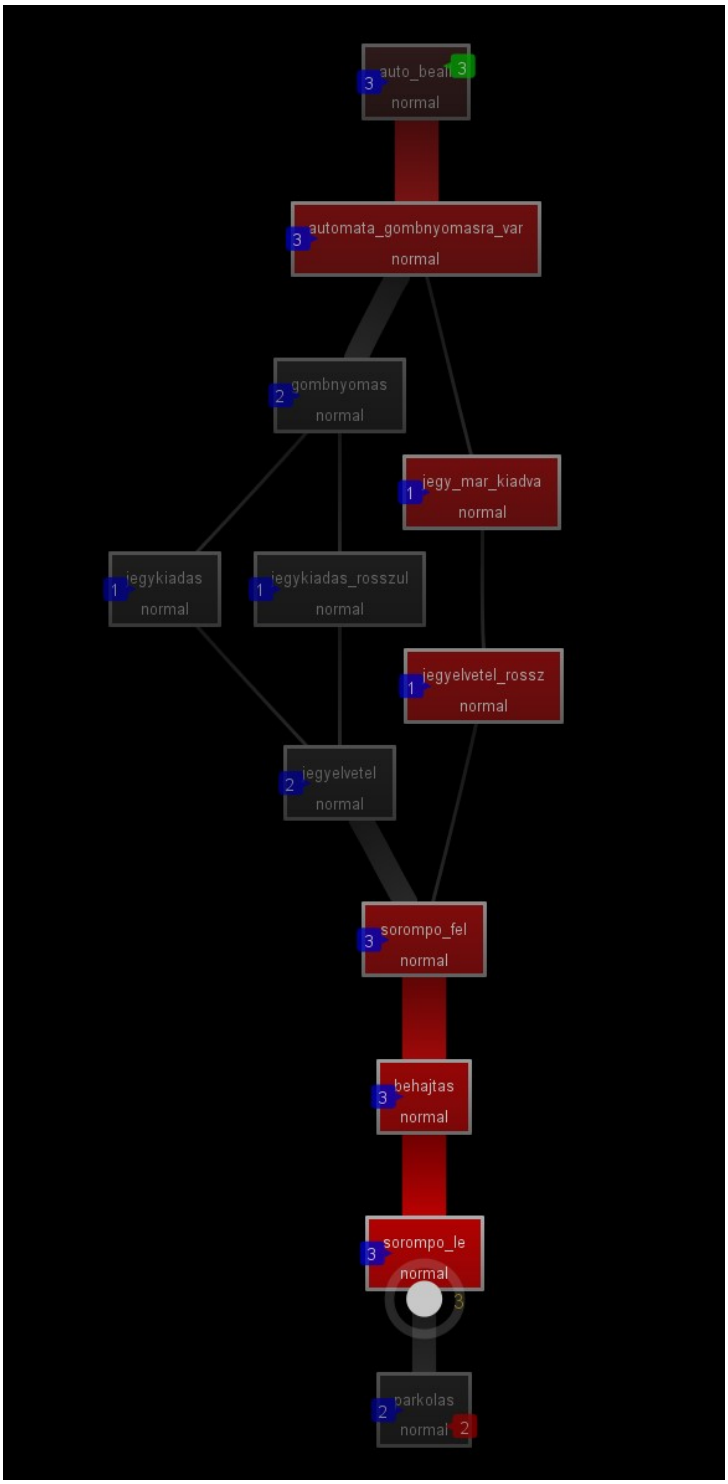
A **„Summary”**, az eddigi adatokat összegzi és megegyezik a klasszikus analízis opcióval. Felsorolás jelleggel táblázatba foglalva láthatjuk az eseménynapló legfontosabb adatait. Ezt az ablakot minden adattal együtt lehetőségünk van HTML fájlként is elmenteni, mindössze a jobb felső sarokban lévő „save HTML” gombra kell kattintanunk.
Elsőként az α -algoritmust fogjuk lefuttatni, melynek eredményét a 13. ábra mutatja.
Tudjuk, hogy három részfolyamat van megjelenítve egyetlen ábrán, így helyenként a párhuzamos elrendezés valójában csak látszólagos. Először lefut a hibamentes első eset, majd az első hiba, végül pedig a második hiba. Az ábra kétség kívül jónak ígérkezik, azonban létezik egy plugin, amivel még könnyebben meg tudjuk állapítani, hogy a folyamat az szerint fut le, ahogy a log-ban felépítettük. Ez a korábban is használt „Fuzzy Miner” algoritmus.
Megnézzük még az α-algoritmus által generált Petri-hálóra a „Fuzzy Miner” animációs plugint. Ehhez győződjünk meg, hogy a kimeneti modell az aktív ablak. Kattintsunk a „Mining” menüpontra, majd a „Selected Petri net” opció kiválasztásával keressük meg a listából a „Fuzzy Miner” plugint. Kattintsunk az előugró ablak jobb alsó sarkában lévő „start mining” feliratú gombra. Az ezt követő ablakon kattintsunk a legutolsó „Animation” fülre. A „Select log for animation” résznél válasszuk ki a legördülő listából az α-algoritmus eredmény modelljét (Result – Alpha algorithm plugin on Raw log_hibakezeles.mxml). A „discrete animation (inject timestamps)” opciót ne pipáljuk ki, hiszen minden időbélyegünk helyes, a három eset, azaz a három autó egymást követően fog megérkezni. Ha megvagyunk, nyomjuk meg a „view animation” gombot.
Ekkor egy hasonló ábrát kapunk, mint amit az α -algoritmus generált. A 14. ábrán láthatjuk az animációs modellt. A „Play” gomb megnyomásával azonnal szimulálhatjuk a vezérlejek mozgását. Azt fogjuk tapasztalni, hogy első esetben a hibátlan eset fog lefutni, azaz az első autós a várt körülmények között fog leparkolni. Miután halad tovább az animáció, elérkezünk a második részfolyamathoz, ahol már a gombnyomás után nem a „jegykiadas” esemény fog következni, hanem a „jegykiadas_rosszul”, hiszen az automata itt két jegyet adott ki. A továbbiakban a folyamat szinkronizálódik az első részfolyamathoz hasonlóan a „jegyelvetel” átmenetnél, és a folyamat a normál körülmények között fog végződni, vagyis a sorompó felnyílik, az autós behajt és leparkol. A

harmadik esetben – ami a 14. ábrán éppen szerepel – a második hibaestre vonatkozó részfolyamat fut le. Itt az előző hibából fennmaradt jegyet fogja a sofőr elvenni, gombnyomás nélkül.



13. ábra – ProM, α-algoritmus kimeneti modellje

A 14-es ábrán, a legelső eseményen szereplő, jobb felső sarokban lévő szám jelenti az éppen aktuális részfolyamat azonosítóját (csakúgy, mint a mozgó, fehér vezérjel melletti szám), míg az események bal felén elhelyezkedő számok azt jelölik, hogy a folyamat lefutása során az egyes átmenetek hányszor tüzeltek.



14. ábra - Hibakezeléses log Fuzzy Miner plugin animációs modellje