Segédlet a Rendszermodellezés (VIMIAA00) házi feladathoz

Hibatűrő Rendszerek Kutatócsoport

2019

Tartalomjegyzék

Bevezetés

1. Előszó

Jelen segédanyag a BME VIK elsőéves informatikus hallgatói számára készült a *Méréstechnika és Információs Rendszerek Tanszéken* Lucz Soma és Farkas Rebeka munkájának felhasználásával, és a Rendszermodellezés (VIMIAA00) című tárgy házi feladatainak elkészítésében segít. A Yakindu eszköz[1] egy állapot alapú modellezést, szimulációt és kódgenerálást támogató eszköz.

Figyelem! A szöveg a *Yakindu* 3.5.1-es verziójával van összhangban. Nyomatékosan kérjük, hogy idén (2019-ben) a házi feladat elkészítéséhez is ezt a verziót használják, mert más (akár újabb, akár régebbi) verziókkal kompatibilitási probléma léphet fel!

[1] `http://statecharts.org/`

2. Alapismeretek

A Yakindu modellező eszköz az Eclipse nevű fejlesztőkörnyezet egy kiterjesztéseként érhető el. Mindkét szoftver nyílt forráskódú és az *Eclipse Public Licence* keretében ingyen használható. Ez a fejezet egy áttekintést ad az Eclipse és a Yakindu főbb fogalmairól.

2.1. Eclipse bevezető

Az Eclipse egy ingyenes, nyílt forráskódú és többcélú fejlesztőkörnyezet, amely egy közös platformból és arra épülő *plugin* (beépülő) modulokból áll. A megfelelő pluginek kiválasztásával ill. saját pluginek fejlesztésével az Eclipse képességei szabadon megválaszthatóak. A legtöbben Java fejlesztőkörnyezetként találkoznak vele (A programozás alapjai 3 tantárgy), pedig többféle előre csomagolt változata is létezik (pl. C/C++ fejlesztéshez, webfejlesztéshez, modell alapú szoftvertervezéshez stb.), melyek mindegyike a célnak megfelelő plugineket tartalmazza.

2.2. Az Eclipse munkaterület

Az Eclipse a munkát *Workspace*-ekbe (munkaterület, munkakönyvtár) szervezi, amely a merevlemezünk egy erre célra kijelölt (de szabadon megválasztható) könyvtára. A workspace-ek *Project*-eket tartalmaznak, amelyeket szükség esetén *Working Set*-ekbe lehet szervezni. Többféle projektet lehet létrehozni, pl Java, C++, Plug-in projekt stb. Projekteket lehet exportálni (pl. `zip` fájlba tömörítve), illetve importálni, így egy projekt workspace-ek és számítógépek között hordozható.

2.3. A Eclipse felhasználói felület

Első indításkor az Eclipse megkérdezi, melyik workspace-ben szeretnénk dolgozni. Adjunk meg neki egy mappát (ha nem létezik, majd az Eclipse létrehozza). Figyeljünk, hogy a mappa elérési útjában ne szerepeljenek ékezetes karakterek, vagy szóközök. Pipáljuk ki, hogy ne kérdezze meg minden indításkor (indulás után bármikor lehet workspace-t váltani). Ez a workspace lesz a későbbiekben a projektfájlok helye, ide fog dolgozni alapesetben a program.

Megjegyzés. Természetesen máshonnan is linkelhető projekt a workspace-be másolása nélkül, de a workspace könyvtár használata kényelmes megoldás arra, hogy egységes helyen tároljuk az aktuális munkáinkat.

Amíg az új workspace-ben nincsenek projektjeink, az Eclipse megjelenít egy üdvözlőképernyő fület. Ezt nyugodtan bezárhatjuk.

Az Eclipse elindítása után a *Workbench* felület látszik, amely nyitott szerkesztőkből és nézetekből, valamint menüsorból, eszköztárakból, állapotsorból áll.

A *szerkesztőket* (Editor) használjuk a munkaterület fájljainak megnyitására és módosítására. Többféle szerkesztő használható egyszerre akár több példányban – pl egyszerre több Java fájl lehet megnyitva, vagy akár Java és XML szerkeszthető együtt. A szerkesztőterületek igény szerint különféleképpen elrendezhetőek a képernyőn, akár külön ablakba is áthúzhatóak.

A szerkesztők mellett további funkciókat tesznek elérhetővé a különféle *nézetek* (View). Például a **Console** nézet a fejlesztőkörnyezetből elindított programok szabványos be- és kimeneti (ill. hibakimeneti) konzolját mutatja, a **Search** nézet a kereséseink találatait listázza stb. Talán a legfontosabb nézet a **Project Explorer** (valamint **Package Explorer** stb.), amelyen keresztül a workspace tartalmát böngészhetjük.

A nyitott nézetek, eszköztárak stb. körét és elrendezését az éppen használt *perspektíva* (Perspective) határozza meg, amelyet valamely munkafázis támogatására válogattak össze. A perspektíva természetesen testreszabható a nyitott nézetek kézi áthelyezésével, bezárásával, valamint újak megnyitásával (**Window** | **Show View**).

2.4. Yakindu bevezető

A Yakindu egy nyílt forráskódú eszköz reaktív, eseményvezérelt rendszerek specifikálására és fejlesztésére állapotgépek segítségével. Tartalmaz egy könnyen használható grafikus szerkesztőt, eszközöket

validációhoz és szimulációhoz, valamint kódgenerátorokat különböző platformokra. A Yakindu Eclipse pluginek csoportjaként van megvalósítva, saját perspektívával, szerkesztőkkel és nézetekkel, amelyeket a következő fejezetek tárgyalnak.

3. A modellező eszköz telepítése

A modellező eszköz a Java nevű programozási nyelvre épül (erről részletesebben A programozás alapjai 3 tárgyból lesz szó), ezért szükséges a Java fejlesztési környezet telepítése (3.1. fejezet). Továbbá a Yakindu telepítésének két alternatív módját is bemutatjuk. A két módszer közötti fő különbség, hogy az első módszer (3.2. fejezet) nem igényli licenc igénylését, míg a második módszerhez (3.3. fejezet) licenc kérelmezése szükséges. Utóbbi módszer lassabb, így javasoljuk az előbbi használatát!

3.1. Java környezet

Ahhoz, hogy az Eclipse alapú Yakindu, továbbá a házi feladathoz segítséget nyújtó további eszközök futni tudjanak, a Java fejlesztőkészlet (Java Development Kit, JDK) előzetes telepítése szükséges (minimum 8-as verzió, javasolt 11-es verzió).

A JDK telepítőkészletet az Oracle honlapjáról[2] lehet letölteni. Az oldalon a **Java SE Development Kit** feliratú szürke dobozban az **Accept License Agreement** lehetőséget kell kiválasztani, ekkor az alatta lévő listából letölthető a platformunknak megfelelő telepítőfájl. A telepítés **Next, Next, Finish** típusú, nem igényel különösebb fejtörést. Amennyiben valamilyen további reklámtermék (pl. Ask.com) telepítését is felajánlja, azt nem szükséges elfogadni.

Megjegyzés. A házi feladat megoldásához a hivatalos Oracle JVM-et ajánljuk. OpenJDK-val maga a feladat megoldható, de ez nem tartalmazza a javafx csomagot, amit az opcinálisan használható, kipróbálást lehetővé tévő grafikus felület igényel (ld. 3. szakasz). Ehhez Linux rendszereken az `openjfx` csomag telepítése szükséges.

3.2. A Yakindu telepítése létező Eclipse példányra

Az Eclipse legfrisebb verziója letölthető a szoftver hivatalos oldaláról[3]. A telepítés során az *Eclipse IDE for Java Developers* változatot kell kiválasztani. A telepítés végeztével el kell indítani az Eclipse-t a Yakindu telepítéséhez.

Megjegyzés. A telepítő nem minden esetben regisztrál parancsikont a start menübe. Ha ez nem történne meg, akkor meg kell keresni a telepítés mappáját, majd ott az `eclipse.exe`/`eclipse`/`eclipse.app`-re kattintani.

Miután elindult az Eclipse a felső menüsorban a **Help > Install new Software...** menüpontra kell kattintani. Elsőként hozzá kell adni egy új forrást.

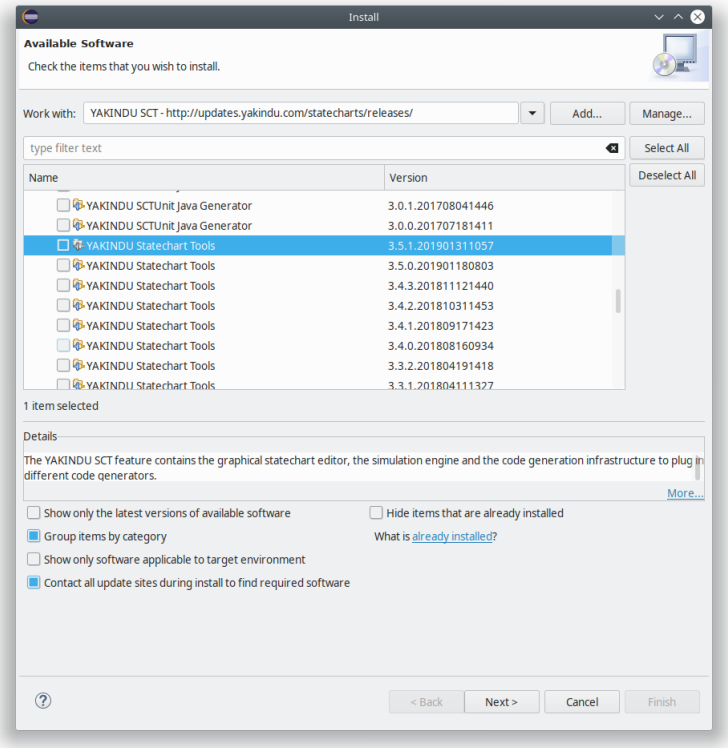
Ehhez a jobb oldalon az **Add...** gombra kell kattintani, majd a felugró ablakban a Name mezőbe beírni, hogy "Yakindu", a Location mezőbe pedig "http://updates.yakindu.com/statecharts/releases/ " (természetesen mindkét esetben az idézőjelek nélkül). Ezt követően az **Add** gombra kell kattintani.

Az új forrás hozzáadása után a *Work with:* mezőben ki kell választani a Yakindu forrását. A kiválasztást követően megjelenik három kategória a telepítő ablakban. A továbblépés előtt alul, meg kell győződni róla, hogy a *Show only the latest versions of availavle software* mező **nincs** bepipálva.

A telepítendő csomagok a **csak és kizárólag** a következőek:

- YAKINDU Statechart Tools Standard Edition > YAKINDU Statechart Tools **3.5.1**-es verzió
- YAKINDU Statechart Tools Standard Edition > YAKINDU Statechart Tools Base **3.5.1**-es verzió
- YAKINDU Statechart Tools Standard Edition > YAKINDU Statechart Tools Java Code Generator **3.5.1**-es verzió

[2] `http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html`
[3] `https://www.eclipse.org/downloads/packages/installer`



1. ábra. Yakindu telepítése az *Install new Software...* menüben

Figyelem! Fontos, hogy csak az előbb felsorolt három csomag legyen kiválasztva, így a YAKINDU Statechart Tools Java Code Generator helyett véletlenül se válasszuk a YAKINDU SCTUnit Java Generator csomagot! Amennyiben több is kiválasztásra kerül, úgy nagy valószínűséggel települni fog függőségként a Licensemanagement komponens, és szükséges lesz licencet igényelni.

A szükséges csomagok kiválasztása után Next, Next, Finish típusú, egyértelmű a telepítés. A telepítés után érdemes ellenőrzini a telepítés sikerességét. Erről további információ a 3.4. fejezetben található.

3.3. A Yakindu telepítése a hivatalos oldalról

A Yakindu legfrissebb verziója letölthető az azt gyártó cég (itemis AG) hivatalos honlapjáról [4], azonban:

Figyelem! A hivatalos oldalról, csak a *legfrisebb* verzió tölthető le, ami előfordulhat, hogy újabb, mint a tárgy által elvárt verzió. A technikai problémák elkerülése végett, a Yakindut a cég repository-jából érdemes letölteni[a], ahol először a **3.5.1-es verziót** kell kiválasztani, majd a megfelelő platformot.

[a] `https://updates.yakindu.com/statecharts/products/standard/`

A letöltés után, a letöltött zip fájlt ki kell csomagolni, majd a benne lévő `SCT.exe`/`SCT`/`SCT.app`-re kattintva lehet elindítani a szoftvert.

A Yakindunak csak bizonyos részei ingyenesek, a haladó funkciók használatához licenccel kell rendelkezni. A hivatalos oldalról letöltött Yakindu verzió tartalmazza a Licensemanagement komponenst, ami a ezek kezelését végzi, és blokkolja a szoftver használatát, amíg a felhasználó meg nem ad egy érvényes licencet.

[4] `https://www.itemis.com/en/yakindu/state-machine/`

A szükséges licensz beszerezhető a Yakindu honlapjáról[5]. Itt a *Professional Edition*, majd az *Academic License* pontot kell kiválasztani. A licenc megszerzéséhez szükséges egy egyetemi email cím, amit a HSZK[6] biztosít. Az ehhez szükséges felhasználónév/jelszó megegyezik az ural2 szerverhez szükséges felhasználónév/jelszó párossal, és lekérhető a BME hallgatói account adminisztrációs oldalról[7]. A licencet erre az email címre fogja elküldeni a cég.

A HSZK-s email cím helyett van lehetőség `@edu.bme.hu`[8], illetve `@sch.bme.hu`[9] cím használatára is, ezekhez azonban további segítséget nem nyújtunk.

Megjegyzés. A licenc importálására a **Window > Preferences** menüpontban van lehetőség, bal oldalt kikeresve a **Yakindu Licenses** ablakot.

Figyelem! Az egyetemi licenckérelmek elbírálását kézzel végzik a cégnél. Emiatt számítani kell egy 2-3 munkanapos átfutási időre, ezért gyorsabb a másik módszer (3.2. fejezet).

Figyelem! A Yakindu licence tiltja a virtuális gépeken futást, így aki virtuális gépen szeretné futtatni, az használja az első telepítési módszert.

A telepítés után érdemes ellenőrzini a telepítés sikerességét. Erről további információ a 3.4. fejezetben található.

3.4. Telepített verzió ellenőrzése

A telepítés után érdemes elvégezni az ellenőrzést, miszerint sikerült-e a megfelelő verziót feltelepíteni, a későbbi problémák elkerülése végett.

Az ellenőrzéshez először meg kell nyitni az Eclipse programot. Ezt követően meg kell nyitni a **Help > About Eclipse IDE** menüpontot, majd az **Installation details** gombra kell kattintani az ablak alján.

Az ezután felugró ablakban megtekinthető az Eclipse példányra telepített pluginek listája. Itt ellenőrizni kell, hogy a listában szerelep-e:

- Yakindu Statechart Tools **3.5.1**-es verzió
- Yakindu Statechart Tools Base **3.5.1**-es verzió
- Yakindu Statechart Tools Java Code Generator **3.5.1**-es verzió

Figyelem! Különösen figyelni kell, hogy a plugin verziók megfelelőek legyenek, a kompatibilitási problémák elkerülése érdekében.

Továbbá érdemes ellenőrizni, hogy a fentieken kívül nem szerelep-e másik telepített Yakindu plugin, mert ez esetben nagy valószínűséggel települ függőségként a Licensemanagement plugin, ami érvényes licenc hiányában zárolni fogja a szerkesztőfelületet. Ez esetben megoldás a felesleges plugin-ek törlése az Eclipse példányból.

4. Projekt létrehozása, importálása

4.1. Yakindu projekt importálása

A házi feladat elkészítése során egy előre megadott projektvázlattal kell dolgozni. Ezt egy `.zip` fájlban adjuk ki, amelyet a következőképpen kell Eclipse-be importálni.

- Katt a **File** menü **Import...** pontjára.

[5] `https://www.itemis.com/en/yakindu/state-machine/licenses`
[6] `https://webmail.hszk.bme.hu/src/login.php`
[7] `https://accadmin.hszk.bme.hu/index.php`
[8] `https://login.bme.hu/office365/`
[9] `https://admin.sch.bme.hu/service_email.php`

– A felügró ablakban a **General** mappán belüli **Existing Projects into Workspace** pont fog kelleni nekünk.
– Itt a *Select archive file* lehetőséget választva adjuk meg a letöltött `.zip` fájl útvonalát. Ekkor az ablak közepén lévő *Projects* részben láthatóvá kell válnia egyetlen projektnek.
– A **Finish** után a Project Explorerben láthatóvá válik az imént importált projektünk. Itt a mappát lenyitva, az **.sct** kiterjesztésű fájlra duplán kattintva elkezdhetünk dolgozni a házi feladattal.

4.2. A modell megnyitása

Nyissuk meg az `.sct` fájlt a Yakindu szerkesztőjében! Az Eclipse ekkor fel fogja tenni a kérdést, hogy *„Perspektívát"* akarunk-e váltani. Ahogy a 2.3. szakaszban írtuk, az Eclipse különböző funkcióihoz az eszközök, nézetek, eszköztárak különféle elrendezései tartoznak, így a Yakindu is olyan elrendezéssel jelenik meg az Eclipse-en belül, ahogy a készítői szerint a legkényelmesebb benne dolgozni. Tehát nyomjunk igent. A perspektíva természetesen később testreszabható a korábban írtak szerint.

4.3. A modell kimentése

Ha elkészült a megoldás, ne felejtsük el elmenteni (Ctrl+S, illetve floppy ikon az eszköztáron). A kész megoldás elmentése után az `.sct` fájlt kell majd tömörítve feltölteni a feladatbeadó portálra (ld. 6. szakasz). Ha a **Project Explorer** segítségével kitallózzuk a fájlt, egyszerű *drag&drop* művelettel kimásolható a workspace-en belüli helyéről.

Egy másik megoldás a számos lehetőség közül, hogy jobb gombbal a fájlt képviselő bejegyzésre kattintunk, és a **Properties** menüponttal megnyitjuk a tulajdonságablakot; ott a **Resource** tulajdonságlap **Location** bejegyzése alatt olvasható a fájl teljes elérési útja, amelyet felhasználhatunk, hogy kikeressül a fájlböngészőben a fájlt.

Ügyeljünk arra, hogy a feladatmegoldás során előállt végleges verziót töltsük fel, semmiképp se az eredetileg kiadott projektvázban található modellkezdeményt.

5. Modellezés, szimuláció

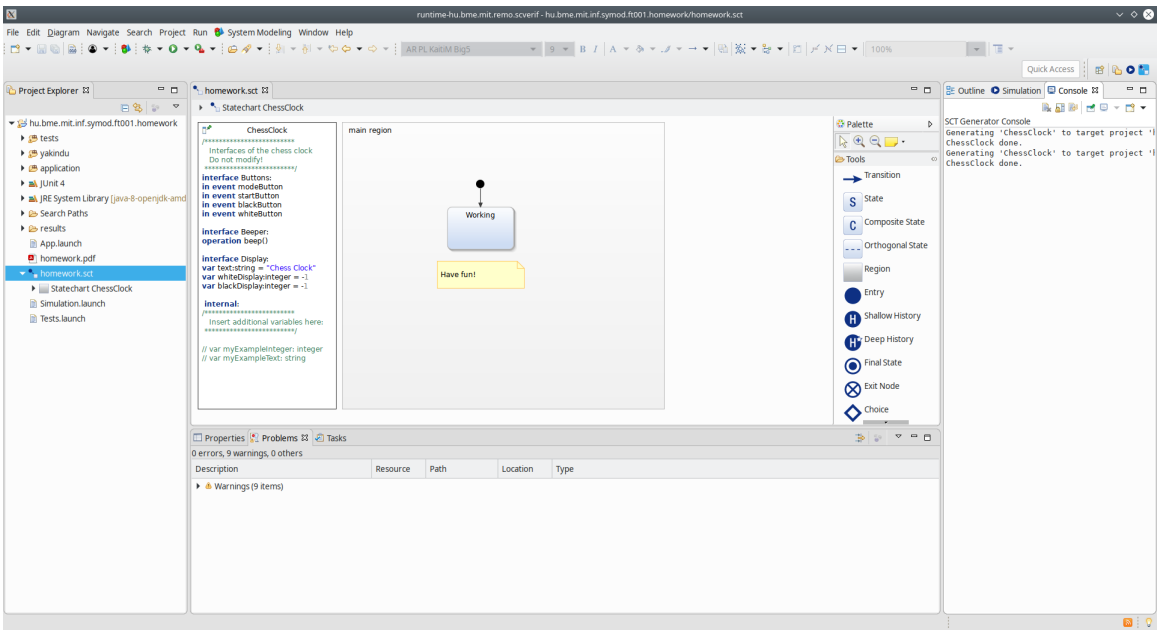
5.1. Állapot alapú modellezés

Az állapot alapú modellezés tágabb témaköréről szóló Rendszermodellezés előadás fóliái elérhetőek a honlapon[10] a többi előadási alkalom fóliáival együtt. A Yakindu modellek az állapotgép (*statechart*) formalizmussal írják le a rendszerek működését. Ezen modellezési paradigmához hasznos elméleti bevezetőt adunk az előadás kiegészítő írásos segédanyagban;[11] ennek az alapos tanulmányozását mindenképpen javasoljuk, mivel a formalizmus elemeinek többségét bemutatja, és Yakindu modelleket használ hozzá illusztrációnak.

5.2. Modellezés Yakinduban

Az **.sct** fájl szerkesztéséhez megnyíló szerkesztőfül három területre van osztva:

– A bal oldali terület szöveges szintaxissal specifikálja az állapotgép nevét és interfészeit (input/output események ill. interfészváltozók neve és típusa).
– A középső területen maga a statechart diagram lesz szerkeszthető. A diagram állapot- és pszeudoállapot-csomópontokból áll, amelyek szürke téglalapként jelölt ortogonális régiókba szervezhetőek. A csomópontok között állapotátmeneti szabályokat reprezentáló élek haladnak amelyek megcímkézhetőek a kiváltó okkal (input esemény vagy időzítés), őrfeltétellel, és végrehajtandó akcióval (változóértékek frissítése, output események).
– A jobb oldali területet a diagramszerkesztéshez használható paletta foglalja el, itt választhatóak ki a diagramra helyezendő modellelemek.



2. ábra. A Yakindu felhasználói felülete

A házi feladat megoldásakor elvárás, hogy az állapotgép külső interfésze az előre kiadottal egyezzen. Így tehát a feladatmegoldás során elsősorban a diagrammal kell dolgozni, **az állapotgép nevének és interfészének módosítása tilos**. Ugyanakkor engedélyezett és ajánlott belső használatú, lokális változókat felvenni; erre a célra fenn van tartva a bal oldali területen egy **internal** megjelölésű szakasz (**Insert additional variables here** megjegyzéssel).

A szerkesztőeszköz részletesebb ismertetésére ezen a helyen nincs mód, de ez a tudás más forrásokból könnyen pótolható. A modellszerkesztő felhasználói felület használatát egy videón mutatjuk be[12], amely lépésről lépésre végighalad a fenti elméleti bevezető segédanyag példáin. A modellező eszköz részletesebb megismeréséhez természetesen rendlekezésre áll a Yakindu honlapján angol nyelven publikált szöveges dokumentáció,[13] továbbá a bemutató videó[14] is.

A videóból nem feltétlenül látszik, de hasznos tudnivaló, hogy a szöveges beviteli helyeken (pl. állapotátmeneti szabályok címkézése) gépelés közben a **Ctrl + szóköz** billentyűkombinációval kódkiegészítéseket (*Code completion*) javasol nekünk a Yakindu; ezzel nagymértékben gyorsítható a munka. További hasznos tipp, hogy a modellezés kezdetekor növeljük meg a *main region* nevű területet, mivel szinte biztos, hogy több helyre lesz szükségünk; a **Ctrl + görgő** kombinációval tudunk kényelmesen nagyítani, illetve kicsinyíteni.

A Yakindu természetesen jóval szélesebb körű felhasználásra alkalmas, mint a Rendszermodellezés házi feladat. Emiatt vannak olyan modellezési elemei, amelyeket más felhasználási esetkere szántak, a házi feladat szempontjából értelmetlenek, és így használatukat nem engedjük meg. Ezen **tiltott elemek** használata nem megengedett, használatuk a megoldás elutasítását vonja maga után. A tiltott elemekről további részletek a 6.3 fejezetben találhatóak.

5.3. A modell működésének szimulálása

Ha már van egy részleges vagy teljes megoldásunk, nyilván szeretnénk meggyőződni annak helyességéről. Ennek egyik eszköze a Yakindu beépített szimulátora.

[10] `https://inf.mit.bme.hu/edu/courses/remo/materials`
[11] `http://docs.inf.mit.bme.hu/remo-jegyzet/`
[12] `https://youtu.be/ev5wEjvje78`
[13] `http://statecharts.org/documentation.html`
[14] `https://www.youtube.com/watch?v=uO6MASCBPrg`

A szimuláció elindításához válasszuk ki a bal oldali Project Explorerben látható Yakindu statechart fájlt (`.sct`) vagy a kiadott projektben erre a célra előkészített `Simulation.launch` nevű „launch configuration" (indítási konfigurációs állományt), majd jobb gombbal kattintva a **Run As**, majd **Statechart Simulation** pontot válasszuk. Így megnyílik a **Simulation View** nézet, és a Yakindu elkezdi szimulálni a modellünket. Piros színnel ki lesz(nek) emelve az állapotkonfiguráció aktív állapota(i), a nagyobb összetett állapotoktól egészen a legszűkebb állapotig.

A jobb oldali **Simulation** fül alatt láthatóak az állapotgép interfészén definiált „változók", illetve „események". A házi feladat példáiban ezek gombok, kijelzők, illetve időesemények lesznek majd. A szimuláció indítása után a *gombok* nyomkodása input eseményeknek felel meg, így ezekkel vezérelhetjük közvetlenül a szimulált állapotgép állapotátmeneteit. A szimulációt a felső eszköztárban lévő piros négyzettel, vagyis a **Terminate** gombbal állíthatjuk le.

5.4. A modell működésének tesztelése

Megoldásunk helyességének ellenőrzésére szolgálnak a kiadott tesztesetek. Ezek lefuttatásával egyszerű helyesírási hibáktól kezdve a bonyolult funkcionális hiányosságokig számos problémára fényt deríthetünk még a házi feladat beadása előtt. Teszteseteink felépítése a következő: először kezdőállapotba állítjuk a vizsgált állapotgépet, majd események sorozatának hatására léptetjük, végül (esetleg közben is) megvizsgáljuk, hogy a feladatkiírásnak megfelelően viselkedett-e. Amennyiben igen, a teszteset *sikeresnek* mondjuk, ellenkező esetben a teszteset *meghiúsul*, és kilistázzuk a hiba okát.

A kötelező házi feladat esetén egy teszteset az alábbi eseményeket tartalmazhatja:

- *Gomb Lenyomása*: a sakkórán lenyomunk egy gombot, amire a rendszernek a feladatkiírásnak megfelelően kell reagálnia.
- *Várakozás:* Idő múlását szimulálhatjuk az állapotgépen. Erre szintén reagálhat az állapotgép.

Események egy sorozata után az alábbi vizsgálatokat tehetjük:

- *Kijelző leolvasása:* A kijelző szövegét betűről betűre összehasonlítjuk egy elvárt értékkel.
- *Játékosok számjelzőinek leolvasása:* A játékosok idejét jelző kijelzőn lévő számot vizsgáljuk.
- *Hangjelzés vizsgálata:* Megnézzük, hogy az ellenőrzés pillanatában elkezdett-e sípolni a sakkóra. (Mivel a modellnek önmagában, az aktuális megvalósításban használt hangjelzések hosszától függetlenül helyesen kell viselkedniük, csak a hangjelzés kezdését mint pillanatszerű eseményt vizsgáljuk.)

A bemelegítő szorgalmi házi feladatban értelemszerűen ennél kevesebb elemből épülnek fel a tesztesetek.

Teszteseteinket a `Tests.launch` állományra jobb gombbal kattintva futtathatjuk, a **Run as** menüponttal. A tesztek hamar lefutnak, két helyen lehet ellenőrizni az eredményeket: a tesztekről szöveges jelentés jelenik meg a konzolon (**Console** nézet), illetve a tesztfuttató keretrendszer (**JUnit**) egy külön erre a célra szánt nézetben is összefoglalja az eredményeket.

A konzolon a sikeres tesztek kódját egyszerűen csak felsoroljuk:

```
test4 Succeeded!
```

Meghiúsult tesztek esetén kiírjuk a vizsgált teszt célját, a beadott eseményeket sorrendhelyesen (azok időpontjával feltüntetve), valamint azt a vizsgálatot, ami meghiúsítja a tesztet. Itt a kapott és a helyette elvárt kimenetet is feltüntetjük.

```
test4 Failed:
Pressing the button three times shows your individual task title FT999 again
----------
- Button at 0s
- Button at 0s
```

```
- Button at 0s
- Failed main display check: expected "FT999" but found "Other text"
```

A **JUnit** nézet felsorolja a teszteket, amelyeket színkóddal (zöld ill. piros) lát el annak megfelelően, hogy sikeres volt-e vagy sem. Az összes teszteset lefutásáról összefoglalót olvashatunk a nézet tetején, valamint egy jelzőcsík is mutatja a futtatás kimenetelét. Egy meghiúsult esetre kattintva megnézhetjük a hiba okát a **Failure Trace** ablakban.

Figyelem! Vannak olyan elemek, amiknek a használatát tiltjuk a feladat megoldása során (részletek: 6.3. szakasz). Ezeknek a használata hibás kiértékelést okozhat a tesztek futtatása során, így kérdéses eredmény esetén érdemes ellenőrizni, hogy használva voltak-e.

5.5. A modell kipróbálása

A részleges vagy teljes megoldásunk helyességét nem csak a Yakindu beépített szimulátorán keresztül próbálhatjuk ki, hanem saját fejlesztésű programba is beágyazhatjuk. (A bemelegítő házi feladat esetén ez a lehetőség nem érhető el.) A kötelező házi feladat esetén el is készítettünk egy sakkórát imitáló grafikus felületet, amelyen keresztül az elkészített modell működése közvetlenül és látványosan kipróbálható. Az alkalmazást az `Application.launch` launch configuration állományra jobb gombbal kattintva indíthatjuk, ha a **Run as** menüpontot választjuk.

Megjegyzés. Olyan modellt nem érdemes kipróbálni, amelyet már a Yakindu szerkesztő is hibásnak mond (egy piros `x` jelenik meg az `.sct` fájl neve mellett). Ezekben az esetekben sokszor az alkalmazás már el sem fog tudni indulni, hibaüzenettel azonnal leáll. Egy másik lehetőség, hogy az alkalmazás elindul, de az állapotgép egy korábbi, még nem hibás felépítése alapján működik.

5.6. Kódgenerálás

A Yakindu képes az állapotgép-modellből olyan forráskódot generálni (különféle programnyelvekben), amelynek működése az állapotgéppel meg fog egyezni. Az állapotgép szerkesztése és elmentése után a Yakindu automatikusan előállítja ill. frissíti a tárgynyelvi forrásfájlokat.

A kiadott projektvázakban már konfigurálásra került a (Java nyelvű) kódgenerálás, amelynek a beállításait egy `.sgen` kiterjesztésű fájl tárolja. Valójában mind a tesztkészlet, mind a sakkóra grafikus felület az így automatikusan előálló forráskódot hívja meg. A házi feladat megoldása során se a kódgenerálás beállításaival, se a generált forráskóddal nem kell foglalkozni, de érdeklődő hallgatók megnézhetik ezeket.

Figyelem! Amennyiben a projektben (szintaktikai) hiba van, a kódgenerátor nem fog lefutni, ami miatt a tesztek a modell egy korábbi állapotát értékelik ki. Ha valami nem működik, érdemes megnézni, hogy a projekt neve mellett található-e egy piros `x`, ami az ilyen jellegű hiba jelenlétét jelzi.

Megjegyzés. Ha valami nem működik, érdemes megnézni, hogy a **Project > Build Automatically** beállítás be legyen kapcsolva a menüben.

6. Feladatkiadás és feladatbeadás

A Rendszermodellezés tágy honlapján[15], a bal oldalt fent látható navigációs fában a **Hírek** rovatra kattintva a tárggyal kapcsolatos aktuális oktatási hírek olvashatóak, amelyekre RSS olvasó segítségével fel is lehet iratkozni. Itt lesz közzétéve a házi feladatokhoz tartozó feladatkiírások illetve a tudnivalók; ezeket a **Házi feladat** aloldalon is összegezzük.

[15] `https://inf.mit.bme.hu/edu/courses/remo`

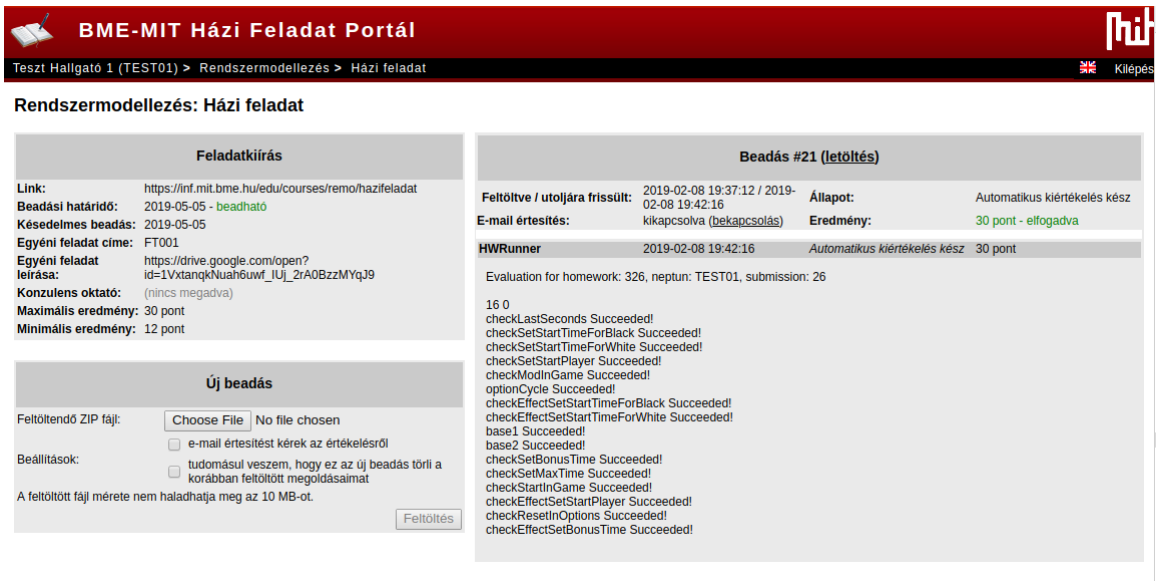
6.1. HF portál használata

A HF portálra elsősorban a **BME címtár** segítségével lehet belépni, de támogatja az egyedi jelszavas bejelentkezést is. Egyedi jelszavat az *jelszó igénylése itt* lehetőség kiválasztásával lehet kérni, amit tanácsolunk, hogy mindenki tegyen meg.

A belépés után ki kell választani a *Rendszermodellezés* tárgyat, ahol lehet látni a tárggyal kapcsolatos kiírásokat. Az egyik házi feladatot (bemelegítő vagy rendes) kiválasztva, meg lehet tekinteni a házi feladatok kezelésének felületét.

A bal oldali panelban a követlező mezők lehetnek érdekesek:

- **Beadási határidő:** A házi feladat leadásának határideje. A határidőt a portál másodpercre pontosan betartja, kérjük erre mindenki legyen tekintettel!
- **Késedelmes beadás:** A pót házi feladat leadásának határideje. **Figyelem!** Technikai okokból, ez csak a házi feladat rendes leadása után néhány nappal kerül kiírásra
- **Egyéni feladat címe:** Minden hallgatónak egyedi házi feladatot generálunk, ez egy azonosító, ami egyértelműen azonosítja számunkra a kapott házi feladatot. Esetleges hibák esetén kérjük ezt az azonosítót mellékelni a leíráshoz.
- **Egyéni feladat leírása:** Erről a linkről lehetséges az egyéni házi feladat letöltése.



3. ábra. A HF portál felépítése

A házi feladatot beadni a bal alsó panel segítségével lehet, míg az eredményekről a jobb oldalon lesz visszajelzés. A beadás során egy `.zip` fájlt kell feltölteni, amiben **csak és kizárólag** egy `.sct` fájl van. Tehát a `.zip` fájl ne tartalmazzon további projektfájlokat (mint `.java` vagy `.sgen` fájlok), illetve az `.sct` fájl a `.zip` fájl gyökerében legyen, ne almappákba ágyazva.

Figyelem! A szerveren korlátos tárhely áll rendelkezésünkre, így nyomatékosan kérjük, hogy az `.sct` fájlon kívül ne legyen semmi az archívumban. Ennek következtében, azokat a megoldásokat, ahol a `.zip` fájlban más is van az elvárt `.sct` fájlon kívül, el fogjuk utasítani.

Figyelem! Új megoldás beadásakor a korábbi megoldás **törlődik** a szerverről, annak visszaállítására nincsen lehetőség.

Figyelem! Vigyázzunk arra, hogy azt a modellt töltsük fel amit szerkesztettünk! A **Copy project into workspace** opció hatására a workspace mappába másolódik a projekt az eredeti letöltés helyéről. Ebben az esetben vagy a workspace-ből töltsük fel a megoldást, vagy másoljuk ki onnan (részletes leírás a 4.3. szakaszban olvasható); de semmiképp se az eredetileg letöltött fájlt töltsük fel változatlan formában.

6.2. Az automatikus kiértékelőről

A tárgyból a házi feladatokat egy automatikusan futó szkript értékeli ki. Ezt a szkriptet igyekszünk rendszeresen, 2-3 naponta legalább egyszer lefuttatni. A pontszámot a kiértékelő állapítja meg, ami a félév végén válik véglegessé.

Mint minden program, így természetesen a kiértékelő is hibázhat. Ennek kompenzálására, a félév végén opcionális védési alkalmakat hirdetünk meg, ahol az erre jelentkező hallgatóknak lehetősége lesz bizonyítani egy oktatónak, hogy a kiértékelő szoftver hibázott. Amennyiben ezt az oktató igazoltnak látja, módosítani fogja a kiértékelő által adott pontszámot.

Figyelem! A beadott házi feladatokra **plágium ellenőrzést** fogunk futtatni. Ezzel kapcsolatban fenntartjuk a jogot arra, hogy utólagosan bárki számára kötelezővé tegyük a védésen való megjelenést, a plágium okozta kételyek eloszlatása érdekében. A védésen az illetőnek a megoldásával kapcsolatos kérdésekre kell válaszolnia, és ezzel meggyőznie az oktatót, hogy a plágiumot ellenőrző szoftver hibázott. Aki erre kötelezve lett, de nem jelenik meg, annak a házi feladata elutasításra fog kerülni!

A kiértékelő szoftver használata egy opcionális segítség, a szoftver folyamatos üzemelésére garanciát nem tudunk vállalni. A kiértékelő előnyeinek kihasználása mindenkinek a saját felelőssége, nyomatékosan javasoljuk, hogy ne az utolsó 1-2 napban próbálják leadni a feladatot.

6.3. Tiltott elemek

A Yakindu természetesen jóval szélesebb körű felhasználásra alkalmas, mint a Rendszermodellezés házi feladat. Így vannak olyan modellezési elemei, amelyeket más felhasználási esetkere szántak, a házi feladat szempontjából értelmetlenek, és így használatukat nem engedjük meg.

A tiltott elemek jelenlétét a kiértékelő szoftver automatikusan ellenőrzi, és amennyiben talál, úgy el fogja utasítani a beadást.

Ezen **tiltott elemek**:

- `always`
- `oncycle`
- Kiváltó esemény nélküli állapotátmenet (a Yakindu a fentiekkel ekvivalensen értelmezi)
- `after 0s` / `after 0ms` / `every 0s` ...
- `[trigger && feltétel]` kiváltó eseményként

Megjegyzés. Az utolsó elem, bár Yakindu-s szemantikával nem számít kiváltó esemény nélküli átmenetnek, a tárgy által tanított szemantikában mást jelent, így nem engedélyezzük a használatát.

Megjegyzés. Az egész tiltás okának mélyén egy állapotgépes modell fizikai megvalósítása szerepel. A Digitális technikából tanult szinkron sorrendi hálózatoknál világos, hogy egy órajel ütésben legfeljebb csak egy szomszédos állapotba tud lépni a rendszer, nem tud átugrani a következőbe. Emiatt viszont az always (after 0s) viselkedését csak úgy tudják megvalósítani, hogy annak a végrehajtását a következő órajelütésre hagyják.

A problémát tovább boncolandó, elő lehet venni a Számítógép architektúrákból tanultakat, miszerint a processzor felváltva hajtja végre az egyes taszkok (szálak) utasításait. Tehát előfordulhat, hogy a A állapotból, belép B állapotba a rendszer, majd nem hajtja végre a C állapotba lépést, ahová egy always-os él mutat B-ből, mert a processzor elveszi tőle a futást, és más feladatot futtat. Így még ha kevés ideig is, de az állapotgép egy olyan állapotban van, amiben **nem szabadna lennie** (mert always miatt azonnal ki kéne lépnie).

Vegyük példának egy atomerőmű irányítórendszerét, ami a különböző szenzorok adatai alapján szabályozza a rudakat. Mivel a fentiek szerinti B állapotban úgysem lesz a rendszer, így nem foglalkozunk azzal, hogy ott megfelelően beállítsuk a modell kimenetét. A tesztelésnél semmilyen probléma nem fog előjönni, mert annak, hogy pont abban a pillanatban vegye el a futás jogát a CPU, az esélye elhanyagolható. Azonban a katasztrófák elkerülése végett nem építhetünk arra, hogy lehetséges, hogy B állapotban van az erőmű, ezt a vonatkozó szabványok is előírják.

6.4. Ismert problémák

A Yakinduban teljes mértékben szabályos használni a `++`/`--` preinkremens, posztinkremens, predekremens, posztdekremens operátorokat. Ugyanakkor a Yakindu modellből Java kódot előállító kódgenerátor, amit a projekt használt, ezekből egy szintaktikailag hibás kódot generál, ami miatt a projekt fordítási hibát fog kapni. Ezért ezek helyett a `+=1`/`-=1` használata javasolt.

Érdemes odafigyelni a sortörésekre az állapotgép készítése során. Amennyiben egy sztringben egy sortörés szerepel (például figyelmetlenségből a végén), azt a kódgenerátor escape-elés nélkül fordítja bele a kódba. Ennek az eredménye egy szintaktikailag hibás Java kód lesz, ami fordítási hibát generál.

Ezek a hibák a gyártó által írt kódban van, így ezek javítását nem tudjuk elvégezni. Várhatóan a félév folyamán nem is fog javítás érkezni.