

# Technikai segédlet a Rendszermodellezés (VIMIAA00) házi feladathoz

Hibatűrő Rendszerek Kutatócsoport

2017

## Tartalomjegyzék

<b>1. Előszó</b>	<b>1</b>	4.3. A modell megnyitása . . . . .	4
<b>2. Alapismeretek</b>	<b>1</b>	4.4. A modell kimentése . . . . .	4
2.1. Eclipse bevezető . . . . .	2	<b>5. Modellezés, szimuláció</b>	<b>4</b>
2.2. Az Eclipse munkaterület . . . . .	2	5.1. Állapot alapú modellezés . . . . .	4
2.3. A Eclipse felhasználói felület . . . . .	2	5.2. Modellezés Yakinduban . . . . .	4
2.4. Yakindu bevezető . . . . .	2	5.3. A modell működésének szimulálása	5
<b>3. A modellező eszköz telepítése</b>	<b>2</b>	5.4. A modell működésének tesztelése .	6
3.1. Java környezet . . . . .	2	5.5. A modell kipróbálása . . . . .	7
3.2. A Yakindu letöltése és beüzemelése	3	5.6. Kódgenerálás . . . . .	7
<b>4. Projekt létrehozása, importálása</b>	<b>3</b>	<b>6. Feladatkiadás és feladatbeadás</b>	<b>7</b>
4.1. A projekt és modell létrehozása . . .	3	<b>Irodalomjegyzék</b>	<b>8</b>
4.2. Yakindu projekt importálása . . .	4		

## 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 kötelező házi feladatának elkészítésében segít. A Yakindu eszköz<sup>1</sup> egy állapot alapú modellezést, szimulációt és kódgenerálást támogató eszköz.

**Figyelem!** A szöveg a *Yakindu* 2.9-es verziójával van összhangban. Kérjük, hogy idén (2017-ben) a házi feladat elkészítéséhez is ezt a verziót használják.

### 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.

<sup>1</sup><http://statecharts.org/>

## 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 pluginok kiválasztásával ill. saját pluginok fejlesztésével az Eclipse képességei szabadon megválaszthatóak. A legtöbben Java fejlesztőkörnyezetként találkoznak vele, pedig többféle előre csomagolt változata is létezik (pl. C/C++ fejlesztéshez, webfejlesztéshez, modell alapú szoftvertervezéshez stb.), melyek mindegyike a célnak megfelelő pluginokat 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. Projektet lehet exportálni (pl. zip fájlba tömörítve), illetve importálni, így egy projekt workspace-ek és számítógépek között hordozható.

## 2.3. A Eclipse felhasználói felület

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 áthúzóhatók.

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 [1] segítségével. Tartalmaz egy könnyen használható grafikus szerkesztőt, eszközöket validációhoz és szimulációhoz, valamint kódgenerátorokat különböző platformokra. A Yakindu Eclipse pluginok csoportjaként van megvalósítva, saját perspektívával, szerkesztőkkel és nézetekkel, amelyeket a következő fejezetek tárgyalnak.

## 3. A modellező eszköz telepítése

### 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 7-es verzió, javasolt 8-as verzió).

A JDK telepítőkészletet az Oracle honlapjáról<sup>2</sup> lehet letölteni. Az oldalon a **Java SE Development Kit** feliratú szürke dobozban az **Accept License Agreement** lehetőséget kell kiválasztani, ekkor

<sup>2</sup><http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>

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.** Figyelem: a hivatalos Oracle JVM-et ajánljuk. OpenJDK-val maga a feladat megoldható, de nem tudjuk biztosítani hogy működni az opcionális használatra szánt, a kötelező házi feladat kipróbálását segítő grafikus felület (ld. 3. szakasz).

### 3.2. A Yakindu letöltése és beüzemelése

A linkelt oldalról<sup>3</sup> tölthető le egy teljes Eclipse a Yakinduval kiegészítve. Itt a jobb oldali oszlopban a *Full Eclipse* alatt lehet kiválasztani az oprendszernek megfelelő verziót. (Alternatívaként természetesen lehetséges egy meglévő Eclipse példány kiegészítése a Yakindu pluginjeivel; ez kezdő Eclipse felhasználóknak nem ajánlott, és nem nyújtunk segítséget hozzá.)

A programot nem kell telepíteni – letöltés és kicsomagolás után egy futtatásra kész állományt kapunk, amit csak el kell indítanunk az `eclipse.exe/eclipse.app`-re kattintva.

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.

## 4. Projekt létrehozása, importálása

### 4.1. A projekt és modell létrehozása

Ahhoz, hogy a Yakinduval megkezdhessük a munkát, szükség van egy tetszőleges projektre az Eclipse workspace-ben. A házi feladat támogatására kiadunk megfelelően előkészített projektvázakat, amelyeket csak importálni kell (ld. 4.2. szakasz). Az alábbiakban viszont leírjuk, hogyan kellene enélkül a projektváz nélkül a „nulláról” elindulni.

Az új projekt létrehozásához a **File** menü **New** menüpontján belüli **Project...** pontra kell kattintanunk. Ha Java kódot szeretnénk majd generálni (ami a házi feladat támogatására kiadott segédeszközöknek kelleni is fog), akkor *Java Project* készítésére van szükség, amely nem meglepő módon az *Java* kategórián belül található; egyéb esetben a *General* kategórián belüli sima *Project* is megteszi. A projekttypus kiválasztása és a *Next* gomb megnyomása után megadhatjuk a projekt nevét, majd egészen nyugodtan nyomhatunk rögtön egy **Finish**-t (a további beállításokra most nem lesz szükségünk).

A bal oldali Project Explorerben láthatóak a projektjeink: itt keressük meg a frissen létrehozottat. Ide fogjuk elkészíteni a Yakindu állapotmodellünket. Jobb gombbal kattintva a mappára **New**, majd **Other**. Felugrik egy ablak rengeteg választási lehetőséggel, itt válasszuk a **Yakindu** kategóriát, azon belül pedig a **Statechart Model** pontot. Fájlnévként megadhatunk bármit, csak a `.sct` kiterjesztésre figyeljünk. Aki akarja, természetesen létrehozhat előbb a projekten belül egy mappastruktúrát (jobb gombbal kattintva **New** és **Folder**) a modell elhelyezésére.

<sup>3</sup><http://statecharts.org/download.html>

## 4.2. Yakindu projekt importálása

A házi feladat elkészítése során a fenti lépések végrehajtására nem lesz szükség, mivel egy előre megadott projektvázlattal kell dolgozni. Ezt egy `.zip` fájlban adják ki, amelyet a következőképpen kell Eclipse-be importálni.

- Katt a **File** menü **Import...** pontjára.
- A felugró 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, amelynek a neve tartalmazza a NEPTUN-kódunkat. Ez lesz a miénk.
- A **Finish** után a Project Explorerben láthatóvá válik az imént importált projektünk. Itt a mappát megnyitva, az `.sct` kiterjesztésű fájlra duplán kattintva elkezdhetünk dolgozni a házi feladattal.

## 4.3. 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.

## 4.4. A modell kimentése

Ha elkészült a megoldás, ne felejtjük el elmenteni (Ctrl+S, illetve floppy ikon az eszköztáron). A kész megoldás elmentése után az `.sct` fájlt kell majd 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 pl. a fájlfeltöltési párbeszédablakban.

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<sup>4</sup> 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;<sup>5</sup> 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).

<sup>4</sup><https://inf.mit.bme.hu/edu/courses/remo/materials>

<sup>5</sup><http://docs.inf.mit.bme.hu/remo-jegyzet/>

- 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.

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<sup>6</sup>, amely lépésről lépésre végighalad a fenti elméleti bevezető segédanyag példáin. A modellező eszköz részletesebb megismeréséhez természetesen rendelkezésre áll a Yakindu honlapján angol nyelven publikált szöveges dokumentáció,<sup>7</sup> továbbá a bemutató videó<sup>8</sup> is.

A videóból nem feltétlenül látszik, de hasznos tudnivaló, hogy a szöveges beviteli helyeken (pl. állapotátmeneti szabályok címkézése) gépelés közben a **Ctrl + szőkőz** 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.

**Figyelem!** A Yakindu természetesen jóval szélesebb körű felhasználásra alkalmas, mint a Rendszermodellezés házi feladat. Így vannak olyan modellezési elemei, amelyeket más felhasználási esetkerek szántak, a házi feladat szempontjából értelmetlenek, és így használatukat nem engedjük meg. (Konkrétabban: a Rendszermodellezés házi feladat kontextusában, a *Digitális technika* tárgyából tanult szinkron áramkörökkel ellentétben, nincs meghatározott *órajel* fogalom, így a kifejezetten erre építő Yakindu nyelvi elemek a szimulációban és tesztelés közben nem fognak helyesen működni.) Ezen **tiltott elemek**:

- `always`
- `oncycle`
- Kiváltó esemény nélküli állapotátmenet (a Yakindu a fentiekkel ekvivalensen értelmezi)

### 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.

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 configurationt” (indítási konfigurációs állományt), majd jobb gombbal kattintva a **Run As**, majd **Statechart Simulation** pontot válasszuk. Így megnyílik a **Simulation View** nézet, és a Yakindu elkezd 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

<sup>6</sup><http://youtu.be/ev5wEjvje78>

<sup>7</sup><http://statecharts.org/documentation.html>

<sup>8</sup><https://www.youtube.com/watch?v=u06MASCPrG>

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 *sikeressnek* 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 tesztekéről szöveges jelentés jelenik meg a konzolon (**Console** nézet), illetve a tesztfutató keretrendszer (**JUnit**) egy külön erre a célra szánt nézetben is összefoglalja az eredményeket.

A konzolon a sikeres tesztek kódját egyszerűen csak felsoroljuk:

```
neptun4 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.

```
neptun4 Failed:
Pressing the button three times shows your Neptun code N3PTUN again
-----
- Button at 0s
- Button at 0s
- Button at 0s
- Failed main display check: expected "N3PTUN" but found "Other text"
```

A **JUnit** nézet felsorolja a teszteseteket, 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.

## 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. Ezekben az esetekben sokszor az alkalmazás már el sem fog tudni indulni, hibaüzenettel azonnal leáll.

## 5.6. Kódgenerálás

A Yakindu képes az állapotgép-modellből olyan forráskódot generálni (különbéle 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 beállításai, se a generált forráskóddal nem kell foglalkozni, de érdeklődő hallgatók megnézhetik ezeket.

## 6. Feladatkiadás és feladatbeadás

A Rendszermodellezés tárgy honlapján<sup>9</sup>, 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 és az előre összekészített Yakindu projektvázak letöltésének lehetősége; illetve a **Házi feladat** aloldalon is összegezzük a tudnivalókat.

A megoldás feltöltése a feladatbeadó portálon<sup>10</sup> lehetséges, kizárólag a határidőt megelőző időszakban. A határidőt szigorúan vesszük, a beadási űrlap a határidő elérésekor percre pontosan lezárul. A feladatbeadó portál használata nem bonyolult, de probléma esetén nyugodtan fordulhatunk a használati útmutatót.<sup>11</sup>

**Figyelem!** A megoldás beadásakor egyetlen `.sct` fájl feltöltése szükséges.

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 4.4. szakaszban olvasható); de semmiképp se az eredetileg letöltött fájlt töltsük fel változatlan formában.

Mivel mindenki egyénileg testreszabott házi feladatot kap, egyéni azonosítás szükséges a házi feladat elemeinek letöltésekor és a megoldás feltöltésekor egyaránt. Az azonosítást a BME Címtár<sup>12</sup> szolgáltatása végzi, ezért aki még nem tette meg, mielőbb regisztráljon magának címtár azonosítót. Ha a portál elérésekor nem történik meg automatikusan az azonosítás, akkor a „nincs jogosultság” hibaüzenetet kaphatjuk; ebben az esetben a bal oldali sávban kattintsunk a **BME Címtár belépés** linkre.

<sup>9</sup><https://inf.mit.bme.hu/edu/courses/remo>

<sup>10</sup><https://inf.mit.bme.hu/user/grader>

<sup>11</sup><https://inf.mit.bme.hu/wiki/jegyzokonyv>

<sup>12</sup><https://login.bme.hu/admin/>

## Hivatkozások

- [1] David Harel: Statecharts: A visual formalism for complex systems. *Sci. Comput. Program.*, 8. évf. (1987) 3. sz., 231–274. p. URL [http://dx.doi.org/10.1016/0167-6423\(87\)90035-9](http://dx.doi.org/10.1016/0167-6423(87)90035-9).