# Bevezető

Hibatűrő Rendszerek Kutatócsoport

2019

## Tartalomjegyzék



## Bevezetés

Vajon mennyi programkódot képes átlátni egy programozó? Hány sorból áll a Windows 7 forráskódja?[1] Hogyan lehetséges mégis kézben tartani ekkora szoftverek fejlesztését? Hogyan garantálhatjuk, hogy a végső termék nem lesz hibás, megfelelő lesz a teljesítménye, és a rajta dolgozó mérnökökön kívül később más is képes lesz megérteni és módosítani a programot? Ezekre a kérdésre ad választ a Rendszermodellezés tárgy a komplex rendszerek modellezésének alapvető aspektusainak bemutatásával.

A *modellezés* egy jól bevált technika a bonyolult rendszerek komplexitásának kezelésében. A tárgy célja ezért a hallgatók megismertetése a szoftver- és hardverrendszerek modellezésének eszközeivel és módszereivel. A modellezés alapfogalmainak bevezetése után a modellek két nagy csoportjával, a struktúra-alapú és viselkedésalapú modellekkel foglalkozunk. Szó lesz a modellek fejlesztéséről, illetve az elkészült modellek ellenőrzéséről, különböző felhasználásairól is.

[1] Körülbelül 40 millió sorból [1].

**Példa.** Vállalatunk a kezdeti sikereken felbuzdulva úgy dönt, hogy belép a közösségi taxizás piacára egy saját szolgáltatás indításával. Ehhez ki kell fejleszteni egy nagyméretű, elosztott rendszert, amelynek részét képezi majd a cég belső gépeire telepített központi szerver és a felhasználók okostelefonjain futó mobilalkalmazás, valamint kapcsolatban fog állni egy bank rendszerével is a fizetések lebonyolításához.

A szenior szoftvermérnök a kezdetektől modellalapú fejlesztésben gondolkodik, hiszen a rendszer mérete és összetettsége több szempontból is indokolja a modellek használatát. Először is a tervezés első lépései során nem lenne szerencsés alacsony szintű technikai problémákkal foglalkozni. Egy megfelelően *absztrakt* modell segítségével a részletkérdések kezdetben elhanyagolhatók, így a mérnökök elsőként a legfontosabb jellemzők kialakítására fókuszálhatnak.

Mivel a megcélzott terület a cég számára új, mindenekelőtt a kapcsolódó fogalmak és a köztük húzódó összefüggések feltárása lesz a cél. A rendszer elosztottsága miatt szükség lesz az egyes összetevők (szerver, mobiltelefon, bank, hálózat stb.) kapcsolatainak modellezésére is. Az ezekhez szükséges eszközöket a *struktúra alapú modellezés* nyújtja.

Amint elkészül a rendszer architektúrája, a mérnökök nekiláthatnak az egyes komponensek viselkedésének tervezéséhez. Itt két kapcsolódó feladat is felmerül: meg kell tervezni a rendszer *folyamatait* (pl. autórendelés, regisztráció, egyenleg feltöltése stb.), illetve ennek megvalósításához az egyes komponensek belső *állapotait* és lehetséges állapotváltozásait. Az ezekhez szükséges eszközöket az *viselkedés alapú modellezés* nyújtja.

Természetesen a vezetőség szeretné elkerülni a tervezési hibák miatti többletköltségeket. A szenior mérnök javaslatára a megszokott *tesztelésen* kívül már a rendszer modelljein is *ellenőrzéseket* fognak végrehajtani, így a hibák azelőtt kiderülhetnek, hogy a tényleges implementációt (drága emberi erőforrás felhasználásával) elkészítenék.

A szoftverek hibátlan működése mellett legalább ilyen fontos, hogy a rendszer *teljesítménye* megfelelő legyen. Ha a szolgáltatásunk nem elérhető vagy lassan válaszol, a felhasználók minden bizonnyal a konkurenciához menekülnek majd, a vállalkozásunk pedig kudarcba fullad. Szerencsére a leendő rendszer teljesítménye is jól becsülhető a modellek alapján, ehhez a *teljesítménymodellezés* nyújt majd eszköztárat.

Részben ide tartozik az is, hogy a legtöbb viselkedésmodell *szimulálható*. Ez több szempontból is a hasznunkra válik majd, hiszen egyrészt már a fejlesztés korai fázisában ki lehet próbálni az alapfunkciókat, másrészt a szimulációk eredményét is felhasználhatjuk a teljesítmény becslésére.

Akár a szimulációkból, akár később az elkészült rendszer monitorozásából rengeteg információ is a rendelkezésünkre áll majd. Ahhoz, hogy ezekből használható tudást állítsunk elő, és adott esetben problémákat, vagy a rendszer rejtett tulajdonságait felfedezhessük, a *felderítő adatelemzés* eszköztárát is bevethetjük majd.

Ebben a jegyzetben a tárgyhoz tartozó legfontosabb témákat mutatjuk be. Az alapfogalmak definícióit releváns mérnöki példákkal illusztráljuk és magyarázzuk, illetve gyakorló feladatokkal segítjük a bemutatott módszerek megértését. Ehhez különböző színű keretes írásokat fogunk használni. Amint az előzőekben látszott, a példák és a definíciók keretben kerülnek kiemelésre, míg a kiegészítő megjegyzéseket kisebb betűmérettel szedtük. A jegyzetben szerepelnek majd feladatok is kék háttérrel, valamint hasznos tippek zöld keretben.

**Definíció.** A *definíció* egy fogalom pontos, precíz meghatározása.

**Megjegyzés.** Időnként megjegyzésekkel fogjuk árnyalni a bemutatott anyagot.

**Feladat.** Gyakorlásképp oldja meg az egyes fejezetekhez tartozó feladatokat!

**Tipp.** A feladatokhoz hasonló kérdések előfordulhatnak a számonkéréseken is.

A jegyzethez tárgymutató is tartozik, amely tartalmazza az előforduló kifejezések magyar és angol megfelelőjét, továbbá egyes angol kifejezések brit angol szerinti kiejtését az IPA jelölésrendszerével.

## A jegyzetről

A jegyzet LaTeX nyelvű forráskódja elérhető a `https://github.com/FTSRG/remo-jegyzet` oldalon. Amennyiben a jegyzetben hibát vagy hiányosságot találnak, kérjük jelezzék a GitHubon egy új *issue*[2] felvételével. A javítási ötleteket szívesen fogadjuk *pull requestek*[3] formájában is.

[2] `https://help.github.com/articles/about-issues/`
[3] `https://help.github.com/articles/using-pull-requests/`

# Hivatkozások

[1] Co.Design: Infographic: How many lines of code is your favorite app? `http://www.fastcodesign.com/3021256/infographic-of-the-day/`, 2016.

# Tárgymutató