

# TDK-dolgozat

Fiák Ádám  
2024.



Budapesti Műszaki és Gazdaságtudományi Egyetem  
Villamosmérnöki és Informatikai Kar  
Méréstechnika és Információs Rendszerek Tanszék

# SysML v2 fejlesztési módszertan kiberfizikai rendszerek modellalapú tervezéséhez és analíziséhez

**TDK dolgozat**

Készítette:

Fiák Ádám  
IV. évfolyam

Konzulens:

dr. Molnár Vince,  
egyetemi tanár

2024.

# Tartalomjegyzék

|                                                                                  |           |
|----------------------------------------------------------------------------------|-----------|
| <b>Kivonat</b>                                                                   | <b>i</b>  |
| <b>Abstract</b>                                                                  | <b>ii</b> |
| <b>1. Bevezetés</b>                                                              | <b>1</b>  |
| <b>2. Háttérismeretek</b>                                                        | <b>3</b>  |
| 2.1. Motiváció . . . . .                                                         | 3         |
| 2.1.1. Modellalapú rendszertervezés előnyei . . . . .                            | 3         |
| 2.1.2. Szimulációs technológiák előnyei . . . . .                                | 6         |
| 2.1.3. Integrált rendszertervezési és szimulációs technológiák előnyei . . . . . | 9         |
| 2.2. Jelenlegi eszközök . . . . .                                                | 10        |
| 2.2.1. Modellezési nyelvek . . . . .                                             | 10        |
| 2.2.1.1. SysML . . . . .                                                         | 10        |
| 2.2.1.2. SysML v2 . . . . .                                                      | 10        |
| 2.2.1.3. Modelica . . . . .                                                      | 11        |
| 2.2.2. Modellek hordozása és szimulálása . . . . .                               | 11        |
| 2.2.2.1. Functional Mock-up Interface (FMI) . . . . .                            | 11        |
| 2.2.2.2. System Structure and Parameterization (SSP) . . . . .                   | 11        |
| 2.2.3. Szimulációs modell generálása rendszermodell alapján . . . . .            | 12        |
| 2.2.3.1. SysPhS . . . . .                                                        | 12        |
| 2.3. Jelenlegi problémák a területen . . . . .                                   | 12        |
| 2.3.1. Tervezési módszertanok hiánya . . . . .                                   | 12        |
| 2.3.2. Modellek megfeleltetésének hiánya . . . . .                               | 13        |
| 2.3.3. Izolált V&V . . . . .                                                     | 13        |
| 2.4. A SysML v2 alapjainak bemutatása . . . . .                                  | 13        |
| 2.4.1. Elemek és típusok . . . . .                                               | 13        |
| 2.4.2. Fontosabb nyelvi elemek . . . . .                                         | 13        |
| 2.4.3. Elemek és típusok viszonyai egymáshoz képest . . . . .                    | 14        |
| 2.5. Korábbi módszertanok bemutatása . . . . .                                   | 14        |
| 2.5.1. Arcadia . . . . .                                                         | 14        |
| 2.5.2. SysMOD . . . . .                                                          | 14        |
| 2.5.3. OOSEM (Object-Oriented Systems Engineering Methodology) . . . . .         | 14        |
| 2.5.4. Intel CoFluent . . . . .                                                  | 14        |
| <b>3. Platformalapú rendszertervezési és -szimulációs módszertan</b>             | <b>16</b> |
| 3.1. Lehetséges megközelítések . . . . .                                         | 16        |
| 3.1.1. Szimulációs fogalmak bevezetése a rendszermodellbe . . . . .              | 16        |
| 3.1.1.1. Annotáció alapú leképezés . . . . .                                     | 16        |
| 3.1.1.2. Szimulációs modellvetület bevezetése . . . . .                          | 17        |
| 3.1.1.3. Lehetőségek összehasonlítása . . . . .                                  | 17        |

|           |                                                                       |           |
|-----------|-----------------------------------------------------------------------|-----------|
| 3.1.2.    | Szimulációs modell architektúrák . . . . .                            | 18        |
| 3.1.2.1.  | Leképezés a Modelica teljes eszközkészletével . . . . .               | 18        |
| 3.1.2.2.  | FMI kompatibilis leképezés . . . . .                                  | 19        |
| 3.1.2.3.  | Lehetőségek összehasonlítása . . . . .                                | 19        |
| 3.2.      | Modellezési módszertan . . . . .                                      | 20        |
| 3.2.1.    | A fejlesztés során felmerült problémák . . . . .                      | 20        |
| 3.2.1.1.  | A fizikai és logikai modellek elválasztásának hiánya . . . . .        | 20        |
| 3.2.1.2.  | Komponensek kohéziója a tervezés első fázisaiban . . . . .            | 20        |
| 3.2.1.3.  | Modellstruktúra átláthatósága . . . . .                               | 21        |
| 3.2.1.4.  | Hierarchikus tervezés támogatása . . . . .                            | 21        |
| 3.2.1.5.  | Szimulációs modellek vetületei, variánsainak reprezentálása . . . . . | 21        |
| 3.2.1.6.  | A modellek újrahasználásának megkönnyítése . . . . .                  | 21        |
| 3.2.2.    | A kidolgozott módszertan javaslat bemutatása . . . . .                | 21        |
| 3.2.2.1.  | Alapgondolatok . . . . .                                              | 22        |
| 3.2.2.2.  | A modellek belső szerkezete . . . . .                                 | 23        |
| 3.2.2.3.  | A fejlesztés menete . . . . .                                         | 25        |
| 3.2.2.4.  | A könyvtárak szerkezete és használata . . . . .                       | 27        |
| 3.2.2.5.  | Szimulációs modellek használata . . . . .                             | 28        |
| <b>4.</b> | <b>Eredmények értékelése</b>                                          | <b>29</b> |
| 4.1.      | Kvadkopter esettanulmány . . . . .                                    | 29        |
| 4.1.1.    | A rendszermodell . . . . .                                            | 29        |
| 4.1.2.    | A szimulációs modell . . . . .                                        | 30        |
| 4.1.2.1.  | Megkötések nélküli verzió . . . . .                                   | 31        |
| 4.1.2.2.  | FMI kompatibilis verzió . . . . .                                     | 31        |
| 4.1.2.3.  | Szimulációs eredmények . . . . .                                      | 32        |
| 4.2.      | Értékelés a motivációk szempontjából . . . . .                        | 33        |
| 4.2.1.    | Rendszertervezési szempontok . . . . .                                | 33        |
| 4.2.2.    | Szimulációs szempontok . . . . .                                      | 34        |
| 4.2.3.    | A két technológia integrálása szerint . . . . .                       | 34        |
| 4.3.      | Értékelés a felhasználási tapasztalatok szerint . . . . .             | 35        |
| <b>5.</b> | <b>Konklúzió és további munka</b>                                     | <b>36</b> |
| 5.1.      | Összefoglalás . . . . .                                               | 36        |
| 5.2.      | További kutatási és fejlesztési lehetőségek . . . . .                 | 36        |
|           | <b>Irodalomjegyzék</b>                                                | <b>38</b> |

# Kivonat

A kritikus rendszerek – mint például a repülőgépek, atomerőművek, vagy vasúti biztosítóberendezések – tervezése során kiemelten fontos, hogy azok bizonyíthatóan megbízhatóan és hibamentesen működjenek. Ennek oka, hogy a hibás működés nemcsak súlyos anyagi károkat okozhat, hanem akár emberéleteket is veszélyeztethet. Ezeknek a szigorú feltételeknek a teljesítésére léteznek tervezési módszertanok, amelyek kiemelt fontosságú részét képezik a különböző verifikációs és validációs (V&V) technikák. A rendszerek komplexitásának növekedése miatt a hagyományos dokumentumcentrikus módszereket elkezdte felváltani a modellalapú rendszertervezés (Model-based Systems Engineering, MBSE), amely mérnöki modellezési nyelvek használ a rendszerek precíz leírására. Emellett a szakterület-specifikus modellezési nyelveknél már elterjedt a szimulációs technológiák - validációs folyamatokban történő, illetve a fejlesztést segítő - alkalmazása.

A szakirodalomban a modellalapú rendszertervezés és a szimulációs technológiák egyetítésének már számos előnyére rámutattak. Az MBSE területén új lehetőségeket és kihívásokat hoz a megjelenés előtt álló SysML v2 nyelv, amely teljesen új alapokra helyezi a széles körben elterjedt SysML-t. Az újdonságok miatt jelenleg még nincs kiforrott módszertan arra, hogyan érdemes SysML v2 nyelven rendszereket modellezni. Ugyanakkor a nyelv ígéretes funkciókkal rendelkezik, amelyek várhatóan a korábbinál nagyobb mértékben teszik majd lehetővé a szimulációs modellek és technológiák integrálását. Ezt felismerve már javában zajlik a SysPhS névre hallgató szabvány átültetése a SysML v2 nyelvre, ami lehetőség teszi majd, hogy szimulációs modelleket származtassunk a rendszermodellekből.

Jelen dolgozat célja összegyűjteni a legfontosabb aspektusokat a komplex kiberfizikai rendszerek modellezésével és a szimulációalapú analízisek integrálásával kapcsolatban, majd egy ezek alapján kidolgozni egy, a SysML v2 újdonságait kihasználó fejlesztési módszertant. Az eredmények értékelése céljából bemutatásra kerül egy esettanulmány is, mely során a javasolt módszerek egy kvadkopter magasszintű terveinek kidolgozásán keresztül kerülnek bemutatásra. Az eredmények lehetővé teszik komplex kiberfizikai rendszerek modellezés és analízisét az új SysML v2 nyelv segítségével, illetve szimulációs technológiák illesztését a rendszermodellhez és a fejlesztési folyamatokhoz.

# Abstract

Critical systems — such as airplanes, nuclear power plants, or railway signaling equipment — must be designed to function reliably and without errors in a verifiable manner. The reason for this is that malfunctioning can cause not only severe financial damage but also potentially endanger human lives. To meet these strict conditions, the design is guided by methodologies, including essential verification and validation (V&V) techniques. Due to the increasing complexity of systems, traditional document-centric methods have started to be replaced by Model-based Systems Engineering (MBSE), which uses engineering modeling languages for the precise description of systems. In addition, the use of simulation technologies – in validation processes and to support development – has already become widespread in domain-specific modeling languages.

The literature has already highlighted numerous advantages of combining model-based systems engineering and simulation technologies. In the MBSE field, new opportunities and challenges are brought by the upcoming SysML v2 language, which is a complete redesign of its widely used predecessor. Due to the new features, there is currently no well-established methodology for modeling systems in SysML v2. However, the language has promising functionalities that are expected to allow for the integration of simulation models and technologies to a greater extent than before. Recognizing this, the process of adapting the SysPhS standard to SysML v2 is already underway, which will enable the derivation of simulation models from system models.

The goal of this paper is to collect the most critical aspects related to the modeling of complex cyber-physical systems and the integration of simulation-based analyses, and then, based on these, to develop a methodology that leverages the innovations of SysML v2. To evaluate the results, a case study will be presented, in which the proposed methods will be demonstrated through the development of high-level designs for a quadcopter. The results will allow for the modeling and analysis of complex cyber-physical systems using the new SysML v2 language, as well as the integration of simulation technologies with system models and development processes.

# 1. fejezet

## Bevezetés

Ahogy a mérnöki tudományok egyre szerteágazóbbak, a tervezett rendszerek pedig egyre összetettebbek lettek – mind a komponensek, mind az érintett szakterületek számának tekintetében –, megjelent az igény, hogy magukat a rendszereket is külön szakemberek tervezzék, ahogy az egyes alkotórészeket is [5]. Így született meg a rendszertervezés területe, melynek elsődleges feladata a rendszerrel szemben támasztott követelmények azonosítása és teljesülésének garantálása.

Kezdetben ezeket a feladatokat írásos dokumentumok segítségével készítették a rendszermérnökök, akik ezeknek a dokumentumoknak a naprakészen tartásával és az ezek közti hivatkozásokkal garantálták a tervek konzisztenciáját és a rendszer működőképességét. Ahogy a rendszerek komplexitása tovább emelkedett, ez a megközelítés kezdett tarthatatlanná válni. Egyre gyakoribbá váltak a tervezési hibák és egyre nagyobb terhet jelentett a mérnökök számára a rendszer állapotát tükröző dokumentumok naprakészen tartása. Erre a problémára nyújtott megoldást a modellalapú rendszertervezés megjelenése, ahol a folyamat központi eleme egy matematikailag is formalizált rendszermodell, amely a számítógépek számára is értelmezhető, ezzel meggyorsítva, egyszerűsítve és gyakran automatizálva olyan feladatokat, melyek korábban komoly terhet jelentettek a rendszermérnököknek és számos hibalehetőséget okoztak a tervezési folyamatban.

Mára a terület nagyon sokat fejlődött és szorosan összefonódott a kritikus rendszerek – mint például a repülőgépek, atomerőművek, vagy vasúti biztosítóberendezések – tervezésével. Ennek elsődleges oka, hogy ezeknek a rendszereknek a fejlesztése során elengedhetetlen, hogy minden egyes követelmény teljesüljön és nyomonkövethető legyen. Azért fontos ezzel külön rendszerszinten is foglalkozni, mert tökéletesen működő alkatrészekből is lehet hibás rendszert építeni, mivel az nem egyszerűen alkatrészek összessége, működését az egyes komponensek megfelelő együttműködése garantálja.

Nem meglepő tehát, hogy a rendszertervezés területén is megjelentek a különböző validációs és verifikációs technikák, melyeknek célja, hogy a megfelelő rendszert és a megfelelő módon tervezzük meg.

Az utóbbi években egyre több figyelmet kapott a rendszertervezők részéről a szimulációs technológiák, mint validációs és verifikációs eszköz felhasználása a rendszertervezés területén [11]. Ezeket a technikákat már évtizedek óta használják ilyen célra más műszaki területeken, ezentúl számos esetben sikeresen használták fel őket a fejlesztés felgyorsítására és gazdaságosabbá tételére is.

Ennek hatására számos munka született az elmúlt években a szimulációk potenciális előnyeiről a rendszertervezés területén és számos további lehetőséget azonosítottak, melyek a szimulációs technikák és a modellalapú rendszertervezés integrációjával további előnyöket biztosítanának a rendszermérnököknek. Mindennek ellenére a két terület integrációjára még nem született széles körben elterjedt, szabványos megoldás.

Ezzel párhuzamosan a szabványosítás utolsó fázisába lépett a modellalapú rendszertervezésben legelterjedtebb SysML nyelv teljesen új, korszerűsített változata a SysML v2 [2]. Az új nyelv nagyon sokat fejlődött az elődjével kapcsolatos tapasztalatokból. A teljesen új, rendszertervezésre szabott metamodellel<sup>1</sup> túl számos új elemmel bővült melyeknek célja a külső források kezelése, a követelmények nyomonkövethetőségének további javítása és a fejlesztőkörnyezetek és egyéb automatikus eszköztámogató szoftverek felé biztosított szabványos interfészek.

Ehhez a merőben új nyelvhez új fejlesztési módszertanok kidolgozására is szükség van. A SysML-hez készült módszertanok esetében többről bejelentették már, hogy készül a SysML v2-re átültetett verzió, azonban ezek sajnos még nem használják majd ki az új szabvány által nyújtott gazdag új eszköztárat. Azonban hosszú távon olyan módszerekre van szükség, amelyek kihasználják ezeket a lehetőségeket, hiszen különben a bennük rejlő lehetőségek kiaknáztatlanok maradnak. Persze ehhez a módszertanokon túl eszköztámogatásra is szükség lesz, de először a modellezési módszerek területén kell megalapozni ezeket a további fejlesztéseket.

Ennek a dolgozatnak a célkitűzése egy ilyen, az új nyelvi elemeket kihasználó módszertan alapjainak kidolgozása. További cél, hogy a javasolt módszereket kezdetettől fogva a szimulációs technológiák figyelembevételével, azokkal összhangban legyenek kidolgozva. Ez hosszútávon megoldást jelenthetne a fent említett integrációs problémákra. Ennek a célnak az elérése jóval könnyebb a SysML v2 nyújtotta új módszerek segítségével. További segítséget nyújt továbbá, hogy egy alapoktól szimulációkkal együtt tervezett módszertan kidolgozása jóval egyszerűbb feladat, mint összhangba hozni olyan módszereket a technológiával, amelyek kidolgozása során még nem volt szempont egy ilyen lehetőség kihasználása.

Ez a dolgozat egy ilyen fejlesztési módszertannal szembeni követelmények és a megvalósítás lehetséges irányait gyűjti össze, majd értékeli ki.

Ezután bemutatok egy az összegyűjtött adatokra támaszkodva megtervezett SysML v2-t használó módszertan alapjait, amely a rendszermodellek új szabvánnyal való tervezésén túl a szimulációs technológiák integrálására is alkalmas.

A kidolgozott módszertan javaslatot ezután egy esettanulmány keretein belül egy kvadkopter rendszermodelljének és szimulációs modelljeinek elkészítésére használom fel, majd ezen keresztül mutatom be az így gyűjtött tapasztalatokat, valamint a módszerek nagyobb előnyeinek a gyakorlatban való megjelenését.

A dolgozat végén értékelem a módszertan kidolgozott alapjait, a tervezési folyamatban nyújtott előnyeik, valamint a rendszertervezés, szimuláció és ezek integrációja kapcsán elért eredményeit.

Azokban az esetekben, ahol a javasolt módszerek csupán előkészítik további eszközök használatát, és csak az ezekkel való kompatibilitást valósítják meg, ott összefoglalom, hogy milyen további kutatási és fejlesztési irányok szükségesek a két terület integrációjának megvalósításához.

---

<sup>1</sup>A nyelv belső szerkezetét és működését leíró modell.

## 2. fejezet

# Háttérismeretek

Ebben a fejezetben bemutatásra kerülnek modellalapú rendszertervezés, valamint a szimulációs technológiák előnyei és a motiváció ezeknek az eszközöknek az egyesítésére. Ezután a területek aktuális problémáinak részletes bemutatása következik.

### 2.1. Motiváció

A szimulációs technológiák már hosszú ideje fontos szerepet játszanak a fejlesztési és tervezési folyamatokban, míg a modellalapú rendszertervezés folyamatosan terjed az iparban, fokozatosan felváltva a klasszikus, dokumentum alapú módszereket. A továbbiakban összegyűjtésre kerültek a technológiák külön-külön nyújtott előnyei, valamint a kombinálásukkal elérhető további előnyök, amelyek a dolgozat motivációját adják [12, 2].

#### 2.1.1. Modellalapú rendszertervezés előnyei

A rendszertervezés elsődleges célja, hogy a egy adott műszaki fejlesztés végén az elkészült rendszer működőképes legyen és be tudja tölteni a rendeltetési célját [2, 5]. Ide tartozik a követelmények összegyűjtése és kezelése, a probléma dekomponálása kisebb egységekre, amelyek egy-egy komponensként megvalósíthatóak konkrét fejlesztőcsapatok által, valamint a különböző szakterületek igényeinek és munkájának az összehangolása. Ezt a feladatot hagyományosan a fejlesztés részletes, írásos dokumentációjával oldották meg, ahol a fejlesztőcsapatok és a rendszertervezők egymás dokumentumaiból tájékozódtak, döntéseiket az aktuális verziókra hivatkozva indokolták.

Más műszaki területekhez hasonlóan, a rendszerek komplexitásának növekedésével a rendszertervezésben is nehézkessé vált a klasszikus módszerekkel történő munka, mivel a rengeteg, gyorsan változó dokumentum mellett nehezzé vált a követelmények teljesítésének nyomonkövethetősége, valamint az egymással kapcsolatban álló komponensek kompatibilitásának garantálása.

Egy nagy komplexitású rendszernél, mint például egy hordozórakéta, látható, hogy milyen sok fejlesztőcsapat dolgozik párhuzamosan különböző komponenseken. A dokumentumalapú megközelítésnél minden változás után frissíteni kellett a saját dokumentációt és eljuttatni azt minden más csoportnak, akinek ez befolyásolta a munkáját, hogy ők is módosíthassanak a komponensükön az új információk alapján. Ahol ilyen sok csapat dolgozik együtt, belátható, hogy a beérkező dokumentum mellett, milyen nehéz is lehet egy-egy csoportnak, hogy a terveit naprakészen tartsa, mivel elég egyetlen dokumentumot elkeverni, és az esetleges inkonzisztencia katasztrofális következményekhez vezethet.

Az Ariane 5 hordozórakéta fejlesztése során például a vezérlőszoftver fejlesztőinek figyelmét elkerülte az a részlet, hogy a rakéta megnövekedett tömege miatt új hajtóműveket terveztek, melyeknek nagyobb tolóereje már nem volt ábrázolható a nyolc bites változók-

ban, amelyek tökéletesen megfelelték minden hajtóműhöz, amellyel a meglévő vezérlőrendszer korábban használták. Az eredmény az volt, hogy a szoftver csak a mért adatok egyik felét használta fel, így a helyes mérési adatokat hibásan vette át – az egyébként jól működő vezérlőrendszer – és a rakéta a kilövés után lezuhant [8]. A baleset megelőzhető lett volna, ha a vezérlőrendszerben tizenhat bites változókra váltanak, amikor a hajtóművek vezérlésében is megtették ugyanezt, de sajnos ez az apró részlet elveszett a számtalan dokumentum között.

Szintén szemléletes példa amikor a Mars Climate Orbiter mars-szonda tért le a tervezett pályájáról, mivel két fejlesztőcsapat nem ugyan abban a mértékegységben tárolta ugyanazt az adatot, így amikor az átadásra került a szoftverkomponensek között, félreértelmezésre kerültek. Ez az incidens is elkerülhető lett volna, ha bárki észreveszi a mértékegységek eltérését, de sajnos mindkét félnek természetes volt a saját választása és elsiklottak az eltérés felett [19].

A probléma megoldására fejlesztették ki a modellalapú rendszertervezést, ahol a dokumentumok helyett egy matematikai alapokon nyugvó modellezési nyelvben írják le a rendszereket. A módszer előnye, hogy mivel a modell alapja egy matematikai formalizmus, így magán a rendszerterven is végezhetőek különböző műveletek, automatikusan ellenőrizhetőek bizonyos feltételek, valamint használhatóak a más területen már bevett verziókezelő-rendszerek a konzisztens állapotok fenntartására. A továbbiakban az új lehetőségek legfontosabb előnyei kerülnek részletesebben bemutatásra [12].

**Nyomonkövethetőség** Mivel az új megközelítésben az egyes modellelemek közt nem csak egy irodalomjegyzékben történő hivatkozás, hanem tényleges, logikai kapcsolat áll fent a modellben, így lehetőség van az egyes követelményeket összekötni azokkal a komponensekkel, amelyek ténylegesen megvalósítják őket [12]. Ezeken a kapcsolatokon keresztül keresni, navigálni lehet a modellen belül és egyszerűen ellenőrizhetővé válik a követelmények teljesítése [2].

Vegyük például a következő követelményt: "A repülőgép több különböző forrásból gyűjt magasságadatokat, a megbízható navigáció érdekében." Könnyen belátható, hogy ez egy kiemelten fontos követelmény a rendszerünk biztonsága érdekében. Ezt a magas szintű követelményt tovább finomítva kapjuk a következőt, amely már inkább a navigációs rendszert fejlesztő szakemberektől érkező rész követelménye az előzőnek. "A repülőgép legalább két, különböző elven működő szenzorral gyűjt magasságadatokat a megbízható működés érdekében." Ez az új részlet azért fontos, mert az eredeti esetben használhattak volna két külön GPS vevő egységet, amely védelmet nyújtott volna az egyik meghibásodása esetén, azonban a GPS rendszer leállása továbbra is teljesen működésképtelenné tette volna a rendszert. Ellenben a pontosabb követelmények mellett, betervezésre kell kerülnön például egy barométer amivel teljesen más elven határozható meg a magasság, így védve mérés módjából fakadó meghibásodásoktól.

A példában, az eredeti magas szintű követelményt összeköthetjük a pontosított követelményekkel, amelyek már egy-egy konkrét, könnyen ellenőrizhető állításból állnak, így, ha valaki azt szeretné ellenőrizni, teljesül-e az eredeti cél, pontos képet kap az ellenőrizendő pontokról. Ha a részletes követelményeket összekötjük a komponensekkel, amelyek az adott részfeladatot megvalósítják, akkor az ellenőrzés is megkönnyíthető.

Ez a nyomonkövethetőség lehetővé teszi lekérdezések készítését, így például automatikusan ellenőrizhető, hogy minden követelményt megvalósítottunk-e valamilyen komponenssel.

**Modellek konzisztenciája** Mint az már korábban is szerepelt, ezeket a modelleket a többi forrásállományhoz hasonlóan lehet verziókezelni például Gittel vagy más verziókezelő eszközökkel [6, 10]. Ezen túl lehetséges garanciákat és feltételezéseket felvenni az adott

komponensekhez [2]. Például a korábbi példában, ahol a változók mérete eltért a két komponensben, megtehető lett volna az adatformátumhoz felvenni a nyolc bites formátumot garanciaként, és ennek a teljesülését feltételezésként a másik komponensben. Így amikor az új hajtómű implementációjában tizenhat bitesre változtatták a formátumot, akkor a garancia megszűnt volna, ami automatikusan hibát jelzett volna a másik komponens modelljében, mivel ott feltételezve volt a korábbi formátum.

**Validáció és Verifikáció (V&V) minden fejlesztési fázisban** Az úgynevezett kritikus rendszereknél, ahol fontos, hogy az elkészült rendszer bizonyíthatóan mentes legyen az implementációs hibáktól, elengedhetetlen a validációs és verifikációs lépések végzése.

A validáció azt mutatja meg, hogy a megfelelő rendszert tervezzük-e, tehát azt, hogy képes betölteni a feladatát, míg a verifikáció feladata bizonyítani, hogy helyesen tervezzük-e a rendszert, vagyis a modellek megfelelnek a követelményeknek és az elvárt tulajdonságoknak. [5]. Például, a validáció feladata, hogy egy repülőgép robotpilótája valóban képes elvezetni a gépet a megadott útvonalon, míg a verifikáció feladata bizonyítani, hogy a feladatot végző program minden – valóban megadható – útvonalon megfelelően működik, például nem okoz meghibásodást ha átrepül az egyenlítő felett, ahol váltania kell északi és déli szélességi körök között; nem kerül végtelen ciklusba; nem szivároztatja a memóriát; nincs olyan állapot amibe beragadhat, például a pilóta bármikor vissza tudja venni az irányítást a robotpilótától, ha szükségesnek ítéli.

A klasszikus rendszertervezésben ezek a lépések csak a fejlesztés későbbi szakaszaiban kezdődhetnek meg, amikor már elkészültek prototípusok az egyes komponensekből [2].

Ezzel szemben, mivel a modellalapú megközelítésnél már a komponensek megvalósítása előtt rendelkezésre állnak különböző részletességű modellek ezekről, amelyeken végezhetőek műveletek, hiába nem fejeződött be – vagy egyáltalán kezdődött el – az implementációjuk.

Így például a fenti robotpilótás példában, ha egy állapotgépes megközelítést választunk, akkor magán az állapotgépen már akkor is lehetséges megvizsgálni, hogy előállhatnak-e különböző veszélyes szituációk, amikor a szoftver és az azt futtató speciális számítógép megvalósítása még meg sem kezdődött.

Ezzel a módszerrel megelőzhető, hogy csak a hosszú fejlesztési folyamat végén derüljön fény koncepcionális hibákra, amelyek az egész rendszer újratervezését és az elkészült prototípus használhatatlanságát vonják magukkal [5]. Ez meggyorsíthatja a fejlesztési folyamatot és védelmet jelenthet a költséges zsákutcák túl későn való felfedezése ellen.

**Modellek újrahasználatossága** A többi műszaki területhez hasonlóan a rendszertervezésben is felmerült az igény a szakemberek részéről, hogy az egyszer elkészített komponenseket minél többször újra lehessen használni más projektek során, ezzel csökkentve a fejlesztési időt és költségeket [2].

Ha egy korábbihoz hasonló rendszert tervezünk, akkor az egész rendszer modelljét szeretnénk felhasználni, más paraméterekkel. Például lehet, hogy egy új autó tervezésekor az nagyobb és gyorsabb lesz, vagy kiegészül plusz biztonsági funkciókkal, de attól még a felépítése, az alapvető funkciói azonosak lesznek egy már elkészült autóéval így sok elemet fel lehet használni egy már kész modellből.

A modellek újrahasználásának másik módja, ha általános komponenseket fejlesztünk. Például precíziós szervomotorokat akarunk gyártani, amelyekbe integrálva van a szabályozásuk is. Ahelyett, hogy minden megrendelő elkészítené a saját – esetleg hibás, vagy pontatlan – modelljét, elkészíthetjük a sajátunkat, amit a megrendelők rendelkezésére bocsátunk, hogy mindenki felhasználhatja azt, amikor a terméküket használja egy másik rendszer fejlesztésénél. Természetesen ez az elv ugyanúgy használható például egy belső

használatra szánt, általános adatgyűjtő modul esetében is, amit más fejlesztéseknél is újra tervezünk használni.

Ezeknek a gondolatoknak a továbbfejlesztése az úgynevezett platform alapú rendszertervezés, ahol először egy általános platformot fejlesztenek egy feladatkör ellátására, majd erre építenek rá konkrét rendszereket.

Ilyen megközelítést használ például több autógyártó is [21]. Van egy alapjuk ami az alvázból, felfüggesztésből és a vezérlőelektronikából áll, és erre építenek autókat más-más motorral, kasznival és különböző a fedélzeti számítógépen szoftveresen megvalósított funkcióval. Ezzel a módszerrel, csak az új alkatrészeknek a modelljét kell elkészíteni, a régit csak elég ezeknek a függvényében újrakonfigurálni, így csökkentve a fejlesztési költségeket. Továbbá mivel számos különböző alváz és elektronika helyett egyetlen általános platformot kell nagy mennyiségben gyártani számos kisebb sorozat helyett, a hasonló termékek sorozatgyártása is jóval gazdaságosabbá válik.

**Automatikusan származtatott nézetek, dokumentumok** Már a dokumentumalapú időkben is megjelentek az egy-egy specifikus terület képviselőinek összegyűjtött összefoglalók és a rendszer egy-egy specifikus tulajdonságát vizsgáló leírások [2]. Ezek eredetileg különálló dokumentumként léteztek, amit ugyanúgy frissíteni kellett a változások szerint, viszont mivel ezek nem feltétlenül egy komponensre, hanem egy nagyobb tervezési szempontra koncentrálnak, így minden érintett alkatrész módosulása új verziót igényelt.

A modellalapú rendszertervezésben a rendszermodell tartalmaz minden információt, ami a fejlesztés szempontjából lényeges. Természetesen a specifikus nézetek továbbra is szükségesek, de lehetőséget kapunk egy jóval egyszerűbb alternatívára, a külön dokumentumok karbantartása helyett.

Ez a megoldás, az összefoglalók rendszermodellből való generálása, amikor szükség van rájuk. Ennek előnye, hogy nem igényel többletmunkát a mérnökök részéről, és bármely pillanatban azonnal rendelkezésre áll a naprakész verzió, nem kell várni, utánakeresni és mindig a tényleges állapotról adnak képet, nem pedig a készítés előtt kiválasztott esetleg inkonzisztens verziók összegzése[12].

A rendszermodell konzisztens állapotának biztosítására már számos megoldás létezik a különböző forrástípusok esetén [6, 10].

### 2.1.2. Szimulációs technológiák előnyei

A szimulációs technológiák már évtizedek óta megjelentek a műszaki tudományok terén, szinte a fejlesztési folyamatok bármely pontjára találhatók példák az alkalmazásukra [5]. Ennek oka, hogy számos előnnyel rendelkeznek, melyek közül a fontosabbak a továbbiakban olvashatóak.

**Megismételhetőség és összehasonlíthatóság** Egy rendszer tervezése és fejlesztése során, rendszerint szükség van a kidolgozott megoldás teljesítőképességének mérésére, vagy az alternatívák összehasonlítására [5]. Ezeknek a vizsgálatoknak az elvégzése, dokumentálása és kiértékelése során két kiemelten fontos szempontot kell szem előtt tartani.

A méréseknek muszáj megismételhetőnek lenniük. A műszaki világban mindennek oka van, és fontos, hogy ez az ok pontosan dokumentálva legyen, arra az esetre, ha később módosításokat hajtanak véget rajta, vagy kivizsgálják valamilyen probléma okát. Ebből következik, hogy nem hivatkozhatunk megismételhetetlen mérési eredményekre, hiszen azok háttere nem vizsgálható meg a későbbiekben.

A megismételhetőség mellett fontos – főleg az alternatívák mérlegelésénél –, hogy két mérés eredményeit ténylegesen össze lehessen hasonlítani. Tegyük fel, hogy egy repülőgépet tervezünk és két hajtómű közül választhatunk. Ha a feladatra mindkettő alkalmas, akkor

valószínűleg a fogyasztásuk lesz a döntő tényező. Ilyenkor dönthetünk úgy, hogy méréseket végzünk az alternatívák megvizsgálásának érdekében. Ekkor azonban adódik egy probléma az összehasonlíthatóság kapcsán.

Ahhoz, hogy érvényes legyen a mérések eredménye, pontosan ugyanazt az utat kellene megtenniük a gépeknek. Ha a tesztpilóta kicsit más útvonalat repül végig, például kicsivel meredekebben emelkedik felszállásnál, más íven kanyarodik, az mind megváltoztatja a végeredményt.

Ha ezt meg is oldjuk például egy robotpilótával, akkor is problémát jelentenek a környezeti tényezők. A levegő hőmérséklete, nyomása, páratartalma, a szélsébség és az esetleges széllekedések mind befolyásolják az eredményeket. Ha ezeket mind figyelembe akarjuk venni, akkor belátható, hogy tényleges repülésből nem lesznek pontosan egyező körülményeink.

A másik lehetőség egy mesterséges tesztkörnyezet, mint egy szélcsatorna, de ott is felmerül a kérdés, hogy tudjuk-e pontosan replikálni az összes tényezőt, illetve, hogy azok minden valós körülménynek megfelelnek-e. Például lehet-e a repüléshez hasonlóan változtatni a légnyomást, hőmérsékletet és páratartalmat, vagy tényleg csak a szél paraméterei lesznek összehasonlíthatóak a mérésekben.

A probléma megoldása lehet, ha szimulációval egy virtuális környezetet és hajtóművet használunk. Mivel ennek a környezetnek minden paraméterét meghatározhatjuk, így garantálható, hogy pontosan olyan hatások érik a két rendszert a tesztek során, valamint az is, hogy a mérések később megismételhetőek lesznek.

**Gazdaságos tesztelés** A fejlesztési folyamatoknak szerves részét képi a kidolgozott megoldások tesztelése, annak megállapítására, hogy azok valóban megfelelően működnek [2]. Azonban a hagyományos tesztek csak a prototípus elkészülése után kezdhetőek meg, ami sokszor költséges és időigényes folyamat [5]. Ennek az elősegítésére hozták létre például a Rapid Control Prototyping azaz a gyors prototípus készítés módszerét a szabályozástechnika területén [22]. A módszer lényege, hogy a tényleges vezérlő legyártása nélkül, szimulált környezetben figyelik az algoritmus működését, így egy megközelítés kipróbálásához, nem szükséges fizikai alkatrészeket legyártani, továbbá nagyobb biztonsággal lehet kísérletezni, mert egy hibás megoldás nem okoz valódi kárt.

Szimulációkkal azonban nem csak gyorsíthatóak a fejlesztési és tesztelési folyamatok. A hagyományos tesztek gyakran igen költségesek lehetnek, a szükséges hardverelemek miatt, főleg ha egy teszt megismétléséhez mindig új alkatrészekre van szükség. Gondoljunk csak egy autótípus engedélyezési folyamata során végrehajtott töréstesztokra. A szimuláció nem spórolhatja meg ezeknek az elvégzését, de van egy előnye az alkalmazásának. A töréstesztet ugyanis addig végzik, amíg a végleges verzió meg nem felel rajtuk. Éppen azért van így, mert máshogy nem garantálható, hogy tényleg megfelelnek a fizikai teszten, minthogy elvégzik azt, az előírások szerint. Ez azt jelenti, hogy – valószínűleg – vannak sikertelen tesztek. Ha ezeknek a verzióknak csak egy részét kiszűrik azzal, hogy egy szimulációban kipróbálják előre, akkor megspórolható az – akkor még egyedi és nem tömeggyártásban lévő – tesztautó legyártásának költsége. Tehát csak akkor kezdenek el teszt példányokat készíteni és töréstesztre küldeni, amikor a koncepció hibákat már kiszűrték, a költséges eljárás nélkül.

Ezen a ponton fontos megjegyezni azonban, hogy a szimulációk soha nem válthatják ki teljesen a fizikai teszteket. Muszáj méréseket végezni a hagyományos eljárásokkal is, hogy meggyőződjünk a virtuális világnak a valóságnak megfelelő működéséről. Természetesen az előbbi részben olvasottak alapján soha nem kaphatunk tökéletes egyezést a felsorolt zavaró tényezők miatt, azt azonban meg lehet állapítani, hogy a modellek kellő pontossággal írják-e le a valós viselkedést és minden fontos tényezőt figyelembe vettünk a szimulációk során.

Ez az összehasonlítási folyamat tekinthető a szimulációs tesztek tesztelésének, ami a többi teszthez hasonlóan nem elhagyható része a fejlesztésnek.

**Szélsőséges esetek tesztelhetősége** Az eddigiekben láttuk, hogy a szimulációkkal hatékonyabban vagy gazdaságosabban végezhetőek el bizonyos feladatok, azonban ez a technológia olyan lehetőségeket is biztosít, amelyek nem valósíthatók meg más módszerekkel [5].

Egy ilyen lehetőség az olyan szituációk ellenőrzése, amelyek másképpen nem tesztelhetők. A tesztelés célja, hogy a rendszer minden részének a megfelelő működéséről meggyőződjünk. Vannak azonban olyan esetek amelyeket a hagyományos módszerrel, kész rendszeren, nem tesztelhetünk.

Vegyünk például egy atomerőművet. A biztonságos működés érdekében elengedhetetlenek olyan biztonsági mechanizmusok, amelyek megakadályozzák a reaktor leolvadását. Ezekben a rendszerekben végezhetünk bizonyos ellenőrzéseket, azonban teljes működés teszteléséhez az kellene, hogy egy reaktort kritikus állapotba hozzunk és megfigyelni, hogy tényleg megfelelően működik-e a rendszer. Ez a módszer nyilvánvalóan nem alkalmazható a gyakorlatban. A tesztek célja az esetleges hibák kiszűrése, hogy elkerüljük a katasztrofális kimeneteket. Nem engedhető meg, hogy az ellenőrzés éppen azt a katasztrófát eredményezze, amelynek a kialakulását hivatott megakadályozni.

Azonban egy virtuális világban, következmények nélkül vizsgálhatjuk ezeknek a mechanizmusoknak a működését is. Természetesen ezek az eredmények nem teljesen egyenértékűek a valós tesztekkel, de sokkal többet jelentenek annál, mintha semmilyen ellenőrzést nem végeznénk, mivel a tervek működését – igazoltan pontos modelleket feltételezve – képesek demonstrálni.

**Hardware és Software in the Loop tesztek (HiL/SiL)** Az előző két részben tárgyalt előnyökhöz kapcsolódnak a HiL és SiL tesztelési technikák [15].

A SiL lényege hogy egy rendszer tesztelni kívánt algoritmusait egy speciális számítógépen futó szoftveres verzióval imitáljuk a rendszer többi része vagy azok modellje felé<sup>1</sup>. Ennek előnye, hogy a vezérlések logikája tesztelhető, anélkül, hogy le kellene gyártani a tesztelendő komponenshez tartozó hardveres elemeket. Ez a technika elsősorban a gazdaságossági szempontból fontos.

Ezzel szemben a HiL teszteknel egy kész komponens köré helyezünk egy virtuális környezetet. Ennek előnye, hogy a tényleges, kész komponens tesztelhető anélkül, hogy a teljes rendszert felépítenénk ehhez, illetve, hogy a virtuális rendszerrel olyan dolgokat is tehetünk, amelyet egy teljes rendszer esetén nem tehetnénk meg. A korábbi reaktoros példa esetében például, ha szimulálunk egy ilyen reaktort a szenzorokkal és a beavatkozó szervekkel együtt, akkor azt megvizsgálhatjuk, hogy a beavatkozószervek vezérlése megfelelően működik, anélkül, hogy a fent tárgyalt veszélyeket vállalnánk. Így növelhető azoknak a komponenseknek az aránya, amelyeket működés közben is teszteltünk.

Olyan kritikus esetekben mint a fenti példa, ezeket a tesztek nem csak egy mérnöki példányon, hanem minden elkészült darabon végrehajtják, hogy garantálják az elkészült alkatrész megfelelő működését. Ennek már nem a tervek, hanem a konkrét példány működésének tesztelése célja.

---

<sup>1</sup>Hagyományosan csak HiL és SiL technikákat különböztették meg, de ma már léteznek finomabb felbontások is, ezekről részletesebben a megjelölt forrásban olvashatunk.

### 2.1.3. Integrált rendszertervezési és szimulációs technológiák előnyei

Látható, hogy a két technológia alkalmazása külön-külön is számos előnyt biztosít a mérnökcsapatoknak, de a kettő integrációja további előnyökkel járhat. A következőkben ezek az extra előnyök kerülnek bemutatásra.

**Munkafolyamatok konzisztenciájának garantálása** Jelenleg lehetséges párhuzamosan használni modellalapú rendszertervezési és szimulációs technológiákat [5], azonban mivel nincs széles körben elterjedt szabványos megfeleltetés a két terület között, így ezeknek a konzisztenciáját a hagyományos módszerekkel, legtöbbször manuálisan kell garantálni. Ha a két terület között kapcsolatot teremtünk, akkor ennek az ellenőrzése is automatizálható.

**Integrált V&V** A többi fejlesztési lépéshez hasonlóan a V&V műveletek is kihatnak a rendszertervezés folyamatára, így azoknak meg kell jelenniük a modell megfelelő részeiben [2]. Mivel a többi műszaki területen elterjedt a szimulációs technikák alkalmazása, így ennek az elősegítésében nagy szerepet játszana a direkt kapcsolat létrejötte [11].

Ez a validációs és verifikációs tevékenység minőségét is elősegítené, mivel a kapcsolat nélkül csak a rendszermodell és a szimulációs modell külön-külön vett vizsgálata lehetséges. Tehát, bizonyíthatjuk, hogy a rendszer működőképes és az hogy a szimulációs modell a vártaknak megfelelően viselkedik, azt azonban nem tudjuk igazolni, hogy a szimulációkban valóban azt a rendszert látjuk, amelyet a rendszermodell leír.

A szimulációk eredménye csak akkor használható a további fejlesztésekhez, ha az tényleg a rendszer terveinek megfelelő működést mutatja. Ezt jelenleg manuális ellenőrzésekkel lehet biztosítani, azonban ez erőforrás-igényes és számos hibalehetőséget visz a tervezési folyamatba, amelyeket a modellalapú módszerek megoldani hivatottak. Ha a két technológiát összekapcsoljuk, akkor ezek automatizálhatóak, amely gyorsabb és biztonságosabb munkafolyamathoz vezet.

**Szimulációs modellek automatikus generálása** A szimuláció célja, hogy a rendszer virtuális verziójának megfigyelhessük a működését, tehát el kell készítenünk a rendszer szimulálható modelljét az erre szolgáló környezetekben. Ez jelenleg manuális folyamat, de ha kapcsolatot teremtünk a két terület között, akkor lehetségessé válna a rendszer szimulációs modelljének, vagy legalább annak vázának az automatikus generálása. Így a szimulációs szakembereknek kevesebb ilyen repetitív rutinfeladattal kell tölteni az idejüket, illetve ez a módszer garantálja a modellnek, vagy annak alapjának a konzisztens a rendszermodellel, nem kell több iterációban, az ellenőrzések alapján, javítani azt [11].

**Rendszer-identifikáció támogatása paraméter-visszavetítéssel** Gyakori probléma, hogy egy rendszernek nem ismertek a különböző paraméterei, amelyek szükségesek a viselkedés pontos leírásához. Ennek a megoldására használják a különböző rendszer-identifikációs technikákat, amelyek lényege, hogy mérési eredményekből határozzák meg egy rendszer dinamikáját [13].

Ha a többi munkafolyamathoz hasonlóan ezek a munkálatok is reprezentálva vannak a modell munkafolyamatokra vonatkozó részén, akkor lehetőség van arra, hogy ezeknek a méréseknek az eredményét a modellbe automatikusan átemeljük és egy következő lépésben frissítsük is ezek függvényében a szimulációs módszereket. Ezt a folyamatot paraméter-visszavetítésnek nevezik, a célja, a szimulációs modellek paramétereinek olyan hangolása, hogy a szimulációk minél pontosabban megegyezzenek a mért értékekkel.

## 2.2. Jelenlegi eszközök

Az egyes komponensek, valamint a teljes rendszerek modellezésére illetve együttes szimulációjának támogatására számos szabvány született, melyek a probléma különböző aspektusaira kívánnak megoldást nyújtani. A területen végzett fejlesztés során ezekre a már meglévő elemekre célszerű építeni. Nemcsak azért, mert így kezelhetőbb méretűvé válik a probléma, hanem azért is, mert ezek a szabványok már megjelentek az ipari alkalmazásokban is és egyre több eszköz támogatja őket.

Az ilyen meglévő szabványokra épített eszközök emiatt könnyebben integrálható a jelenlegi munkamenetekbe, munkakörnyezetekbe. Mivel ezek a technológiák így kiemelt fontosságúak a későbbi munkára nézve, ezért először célszerű röviden áttekinteni ezeket.

### 2.2.1. Modellezési nyelvek

A rendszereket a különböző nézőpontokból egy-egy erre a célra kifejlesztett nyelv segítségével írják le a mérnökök, Mivel ezekkel a nyelvekkel készült modelleket akarunk összekapcsolni, így érdemes ezekkel kezdeni az áttekintést.

#### 2.2.1.1. SysML

A SysML egy diagramokon alapuló modellezési szabvány a rendszermérnökök igényeire szabva [2]. Jelenleg ez a legelterjedtebb modellezési eszköz a területen. Sajnálatos módon a modellben szereplő komponensekhez készült részletes, doménspecifikus modellekkel szabványos módon nem tud kapcsolatot teremteni.

Leírhatóak benne a követelmények, paraméterek, különböző feltételezések, viselkedések, tesztek, de az ezekhez kapcsolódó külső elemekkel az adatcserét és szinkronizációt kézzel vagy külső eszközökkel kell biztosítani.

A modellben lévő adatokra megfogalmazhatunk lekérdezéseket, így sokkal könnyebben kereshetővé és nyomonkövethetővé válik a tervezési folyamat. A nyelvhez lehetőség van szakterület-specifikus kiterjesztések készítésére, így kiterjesztve a képességeit.

#### 2.2.1.2. SysML v2

A SysML új, jelenleg fejlesztés alatt álló változata [2]. A nyelvet új alapokra helyezik<sup>2</sup>, ezen kívül számos új nyelvi elem kerül be a szabványba, amelyekkel jobban nyomonkövethető a tervezési folyamat és gyakorlatilag a modellbe olvastja a dokumentációt is, beleértve például a különböző konfigurációkról hozott fejlesztői döntéseket, ezek alapjául szolgáló méréseket és követelményeket.

A nyelv a meglévő grafikus szintaxis mellé szövegeset is kap, így gyorsítja a modellezési folyamatot és lehetővé teszi a programkódoknál már használt fejlesztést segítő eszközök használatát.

Ezen túl a szabvány elkészítése során kezdettől fogva kiemelt szempont volt a modellek hordozhatósága fejlesztőeszközök között, valamint az automatizált eszközök számára biztosított szabványos API (Application Programming Interface) kialakítása, megkönnyítendő a fejlesztést támogató eszközök készítését és integrációját a munkafolyamatba.

Az újdonságok közül a munkám szempontjából külön kiemelendő két tulajdonság. Egyrészt a szöveges szintaxissal még kényelmesebbé válik a nyelvi kiterjesztések készítése. A másik fontos újítás, hogy lehetőség van külső fájlok hivatkozására a modellben, ezzel egy laza kapcsolatot teremtve a külső, doménspecifikus modellekkel.

---

<sup>2</sup>Új, saját metamodell készül a nyelvhez, amelyben más alapelemeket definiálnak a korábbiakhoz képest.

### 2.2.1.3. Modelica

Ez egy általános modellezési nyelv, amely rendelkezik nyílt implementációval, ezért bárki számára ingyenesen használható [17].

A nyelv deklaratív megközelítést alkalmaz, tehát a komponenstípusok karakteristikáit kell megadnunk, illetve a példányosított komponenseket fel kell paraméterezni és definiálni a köztük fellépő kapcsolatokat. A megközelítés sok esetben sokkal kényelmesebb tervezési folyamatot tesz lehetővé, mint más megközelítések. A nyelv továbbá kiterjedt domain specifikus könyvtárakkal rendelkezik a műszaki tudományok különböző ágainak támogatására.

A munkám során használt OpenModelica eszköz, amely a nyelv nyílt implementációjához tartozó környezet, alpból támogatja az FMI és SSP szabványok (Lásd: 2.2.2) használatát, amely komoly segítséget jelent a technológiák integrálása során.

### 2.2.2. Modellek hordozása és szimulálása

A fent említett nyelvek természetesen nem az egyedüliek a területen, valamint számos domain-specifikus megoldás létezik a rendszerek leírására és modellezésére mind nyelvi, mind fejlesztőkörnyezeti szempontból. Ezért is van szükség olyan megoldásokra, amely a felhasznált technológiáktól függetlenül képesek reprezentálni a modelleket, hogy azok más eszközök számára is értelmezhetőek legyenek. A következőkben bemutatásra kerülnek erre a célra használt technológiák.

#### 2.2.2.1. Functional Mock-up Interface (FMI)

A Functional Mock-up Interface (a továbbiakban: FMI) szabvány egy egységes interface-szel rendelkező csomagformátumot definiál, amellyel egyes komponensek modelljeit lehet becsomagolni, hogy aztán egy közös szimulátorban (a továbbiakban: koszimulátor) összekapcsolhassák őket és ott a komponensekből felépített rendszert lehessen szimulálni [16].

Az FMI egy nyílt szabvány, tehát bárki használhatja és számos eszköz van, amely támogatja a modellek ilyen formátumba való exportálását. Nagy előnye, hogy mivel a koszimulátor csak a csomagok külső interface-ét ismeri, számos különböző eszközzel készült, eltérő típusú modellt lehet felhasználni egy rendszer összeépítéséhez.

Magát a szabványt úgy tervezték, hogy a csomagok adjanak egy olyan interface-t is, amely a különböző, őket használó eszközöknek szól és változatos funkciókat tegyenek lehetővé a metaadatok biztosításától a szimuláció közbeni monitorozásig. Fontos továbbá, hogy a szabvány úgy lett megalkotva, hogy a lehető legtöbb területen alkalmazható legyen, így például beágyazott rendszerekben elérhető erőforrásokkal is használhatóak ezek a csomagok.

#### 2.2.2.2. System Structure and Parameterization (SSP)

Ez a szabvány bár kapcsolódik az FMI-hez, de szerkezetéből adódóan bármilyen hasonló komponensmodell csomagokhoz alkalmazható, ennek ellenére mi a továbbiakban az FMI csomagok (a továbbiakban FMU-k), viszonylatában vizsgáljuk [18].

A probléma, amit megoldani hivatott, hogy bár az FMU-kból felépíthető egy rendszer, amelyet utána szimulálunk, de az FMI szabvány nem definiálja azt, hogyan is lehetne felépíteni ezekből a csomagokból egy rendszert. Éppen ezért az SSP egy olyan exportálható csomagot ír le, amellyel más, a komponenseket modellező csomagok kapcsolata írható le, így felépítve azokból a rendszert, amelyet szimulálni akarunk.

Mivel ezt a szabványt rendszerint az FMI-vel együtt kezelik, ezért hasonló a támogatottsága, illetve a felhasználást végző eszközök felé nyújtott támogatásának képességei is. Megemlítendő továbbá, hogy ez esetben is is nyílt szabványról van szó.

### 2.2.3. Szimulációs modell generálása rendszermodell alapján

Mivel a rendszermodellezés és a szimulációs technológiák összehangolása már évek óta aktív terület, így megjelentek eszközök amelyek a rendszermodellből való szimulációs modell generálást hívatottak támogatni. A következőekben ezek az eszközök kerülnek röviden bemutatásra.

#### 2.2.3.1. SysPhS

A SysPhS-t a Object Management Group Inc. fejleszti [20]. A szabvány jelenleg az általa definiált speciális típusokkal leírt SysML modellekből specifikálja Modelica illetve Matlab/Simulink modelleket generálásának. A szabvány SysML v2-re való átdolgozása jelenleg is folyamatban van. Jelen dolgozat szempontjából azért fontos, mivel felhasználásával létrehozható a kívánt kétirányú kapcsolat rendszermodellektől a szimulációs modellek felé vezető ágának egy része.

## 2.3. Jelenlegi problémák a területen

A fentiekből látható a modellalapú technológiák előnye, valamint bemutatásra került a környezet, amibe egy új megoldásnak illeszkednie kell. Ebben az alfejezetben a területen aktuális problémák kerülnek bemutatásra, mivel a dolgozat célja ezeknek a problémáknak a mérséklése vagy megszüntetése új módszerek kidolgozása révén.

### 2.3.1. Tervezési módszertanok hiánya

Módszertani szempontból két komolyabb probléma van a további munka tekintetében.

Mivel a rendszermodellek és a szimulációs modellek két eltérő reprezentációját adják a rendszereknek, így azok eltérő szerkezetűek. A területek összehangolásához célszerű olyan modellstruktúrákat kidolgozni, amelyek megkönnyítik az együttes alkalmazásukat és elősegíti a különböző szakemberek együttes munkáját ezen az új határterületen.

A másik probléma, hogy mivel ezek az integrációs technológiák még korai fázisban járnak, így nincsenek olyan kiforrott fejlesztési módszerek, mint önmagában a rendszertervezés vagy a szimuláció esetében<sup>3</sup> [11]. Mint az a modellező nyelvek fejezetben is olvasható, a SysML a legelterjedtebb rendszertervező nyelv, amelynek hamarosan egy merőben új, a feladatára még inkább alkalmas verziója jelenik meg.

Éppen ezért a továbbiakban ezzel a nyelvvel foglalkozom a munkám során. Az, hogy a nyelv még nem készült el teljesen, és nagyban eltér az elődjétől, azt is jelenti, hogy a felhasználására ugyanúgy nincsenek kiforrott módszerek, mint az integrációra. Ez azt jelenti, hogy a továbbiakban egy ilyen módszer kidolgozásán is dolgozni kell. Másfelől ez nagyobb flexibilitást enged a további munka során, mivel nem kell már bevett, de számunkra kevésbé megfelelő módszerekhez igazodni. Nem egy olyan módszertanba kell utólag beépíteni a szimulációkat, amely nem lett erre felkészítve, hanem alkotni kell egy újat, amelynek szerves részét képezi.

<sup>3</sup>A jelenlegi rendszertervező módszertanok összefoglalása a 2.5 részben olvasható.

### 2.3.2. Modellek megfeleltetésének hiánya

Bár számos korábbi munka foglalkozott azzal, milyen előnyökkel járna ezeknek a technológiáknak az integrációja (Lásd: 2.1.3), egy ilyen kapcsolat platformfüggetlen létesítésére még nincsen példa [11]. Olyan integráció, amely egy adott cég termékei között létesít hasonló kapcsolatot, a fenti előnyök egy részének érdekében, már létezik, azonban ezek csak néhány konkrét eszközt kapcsolnak össze, nem általánosan használható módszereket biztosítanak [14].

### 2.3.3. Izolált V&V

A megfeleltetéshez hasonlóan a V&V technikák alkalmazásának és munkafolyamatba olvasztásának előnyei már ismertek azonban a gyakorlatban – többek közt a megfeleltetés hiánya miatt – erre még szintén nincs általános megoldás, csupán konkrét eszközökhöz illeszkedő kezdeményezések és általános de részleges megoldások [11].

## 2.4. A SysML v2 alapjainak bemutatása

Jelen dolgozat célja egy SysML v2 alapú módszertan kidolgozása kiberfizikai rendszerek tervezésére. A módszertan javaslat megértéséhez azonban szükséges a nyelv alapszintű ismerete.

### 2.4.1. Elemek és típusok

A nyelvben a legalapvetőbb koncepció a különböző típusok bevezetése és használata, melyhez két fontos fogalom tartozik [2].

**Definition** Így nevezzük egy típus létrehozását. Ilyenkor csak azt adjuk meg, hogy egy adott típusú elem milyen tulajdonságokkal rendelkezik, milyen más elemekkel és milyen kapcsolatban áll. Az így kapott definíciót használhatjuk fel az adott típushoz tartozó elemek létrehozására.

**Usage** Egy típus felhasználása adott kontextusban. Ez történhet definíciók segítségével, vagy létrehozhatunk nevesített típus nélküli elemet is, ha tudjuk, hogy nem lesz szükség több hasonló elemre, ezzel elkerülve a felesleges definíciók írását.

### 2.4.2. Fontosabb nyelvi elemek

A SysML v2-ben néhány alap építőelemet definiáltak, amelyek kombinálásával leírhatóak a rendszerek. Ezekből az absztrakt építőelemekből definiálhatunk saját, konkrét jelentéssel bíró típusokat a modellek felépítéséhez. A legfontosabb típusok a következők:

**Part** Önmagukban is értelmezhető önálló elemek, tipikusan alkatrészek és funkciók leírására.

**Port** Csatlakozási pontok leírására szolgáló elemek. Fontos hogy ez általános elem, bármilyen csatlakozási pont lehet. Például egy USB csatlakozó, de akár a tűzcsap és a tűzoltótömlő végén levő fémgyűrűk is.

**Connection** Két tetszőleges elem közötti kapcsolatot ír le.

**Interface** Speciális kapcsolat amely két Port kapcsolatot írja le.

**Allocation** Speciális, irányított kapcsolat két tetszőleges elem között. Egy logikai elem megvalósításának és felelősségeinek hozzárendelése egy fizika elemhez. Például egy tolatóradarnál a felhasználó figyelmeztetését egy hangszóró valósítja meg.

### 2.4.3. Elemek és típusok viszonyai egymáshoz képest

A különböző elemek és típusok nemcsak felhasználhatják egymást saját maguk leírásához, hanem különböző speciális viszonyban is lehetnek egymással. A nyelvben található ilyen viszonyokból a fontosabbak:

**Specializáció** Egy típus vagy elem specializál egy másikat, ha rendelkezik a másik tulajdonságaival, de valamilyen szempontból szűkít, pontosít azokon. Például egy motorokra vonatkozó általános leírást specializál egy villanymotor, mert ugyanazokkal a funkciókkal rendelkezik, de a konkrét implementációra már megkötéseket tartalmaz.

**Redefiníció** Egy konkrét elem redefiniálható egy azt specializációval, ilyenkor a belső tartalma felülíródik a konkrétabb definíciónak megfelelően.

## 2.5. Korábbi módszertanok bemutatása

Bár a SysML v2-höz még nincsenek tervezési módszertanok, de elődjéhez több is létezik. Ezeket azért érdemes áttekinteni röviden, mert a terület jelenlegi állapotáról ezekből kaphatunk képet, illetve készítőik várhatóan ezeket fogják átemelni a SysML v2-be is.

### 2.5.1. Arcadia

Az Arcadia egy specifikusan a Capella fejlesztőkörnyezethez készült módszertana [3]. A sajátossága, hogy részletesebben felbontja a projektek elején történő követelménymodellezési fázist és a fejlesztés során az egyes lépések kiindulási modelljének generálásával támogatja a fejlesztők munkáját. Ezen kívül alapvetően domain-specifikus kiterjesztésekre támaszkodik, amellyel a felhasználó szabja személyre a módszertant. Ez nagyban megkönnyíti a későbbi munkát, bár bevezetése így valamivel több munkát igényel, mint más módszertanoké.

### 2.5.2. SysMOD

Általános módszertan rendszertervezésre, a hangsúly itt is a követelményeken és a hasonló előkészítő lépéseken van [23]. Ezen kívül a modellek rendszerezésére tartalmaz ajánlásokat és néhány nyelvi kiterjesztéssel nyújt segítséget néhány feladat elvégzéséhez. Ez a módszertan már átdolgozás alatt áll az új szabványhoz, bár a hozzá kapcsolódó kiterjesztések újrainírása valószínűleg hosszabb folyamat lesz.

### 2.5.3. OOSEM (Object-Oriented Systems Engineering Methodology)

Szintén általános módszertan amely a rendszermodellezés management oldalához ad jóval több támogatást [1]. Ezzel együtt azonban valamivel több megszorítást ad a tervezési folyamatra. Ezekből a legjelentősebbek, hogy kizárólat top-down tervezést támogat és szigorú sorrendiség van a logikai és fizikai nézetek között. A módszertan támogatja a V&V technikák reprezentációját a modellben speciális kapcsolatokkal azonban ezek használata nincsen a módszertanba integrálva, csupán lehetőséget biztosít az alkalmazásukra. Ez a módszertan is már átdolgozás alatt áll a SysML v2-höz.

### 2.5.4. Intel CoFluent

Az Intel által kidolgozott módszertan, amely a SysML mellett még két másik modellezési nyelvet használ fel beágyazott informatikai rendszerek tervezésére [9]. Ez a módszertan

sajnos nem alkalmazható általános rendszertervezésre, mivel például mechanikai komponensek modellezésére nem alkalmasak. Cserébe a saját szűkebb területén kiemelkedő képességei vannak. A beágyazott elektronika és szoftver szimulálására támogatást ad, így egy szűk területen biztosítja azokat az előnyöket, amelyek általános rendszerekhez való illesztésének elősegítése jelen dolgozatnak is a témája.

## 3. fejezet

# Platformalapú rendszertervezési és -szimulációs módszertan

Ebben a fejezetben kerülnek bemutatásra a téma kapcsán létrejött saját eredmények. Először a megoldás lehetséges módjai kerülnek bemutatásra az előnyeikkel és hátrányaikkal együtt. A második alfejezet a rendszerek SysML v2-ben történő modellezéséhez létrehozott módszertant mutatja be, amelybe beleépülnek a szimulációs technológiákhoz létrehozott módszerek. Végül egy konkrét esettanulmányon keresztül bemutatásra kerülnek a leírt módszerek.

### 3.1. Lehetséges megközelítések

A modellalapú rendszertervezési és a szimulációs technológiák integrálása módosításokat követel mind a két oldal modelljeiben. Ezek a módosítások és tervezési módszerek a két terület közötti kapcsolat, egy kölcsönös megfeleltetés miatt elengedhetetlenek. Azonban több különböző lehetőségünk van, ennek a megvalósítására. Ez az alfejezet ezeket a megközelítéseket veszi sorra és hasonlítja össze.

#### 3.1.1. Szimulációs fogalmak bevezetése a rendszermodellbe

A rendszermodell tekintetében a leglátványosabbak ezek a változások, mivel itt ténylegesen meg kell jelenni a másik szakterületnek, hogy az beolvadhasson a fejlesztési folyamatba. A szimulációs fogalmak bevezetésére két módszert dolgoztunk ki, amelyeket részletesen tárgyalunk majd.

##### 3.1.1.1. Annotáció alapú leképezés

Az új fogalmak és tulajdonságok bevezetésére a legkézenfekvőbb megoldás a SysML v2 nyújtotta lehetőség különböző metaadatok bevezetésére. A megoldás lényege, hogy létrehozunk egy importálható fájlt amely minden fontos fogalomhoz tartalmaz egy-egy annotációt, amivel az adott elem bevezethető. Gyakorlatilag kézzel megcímkézzük, hogy melyek a fizikai és logikai elemek, melyik kapcsolat milyen típusú, stb.

Ezen kívül lehetőség van szabályokat megadni, amelyek segítenek megakadályozni a hibás használatot, például nem jelölhetjük külön szimulációs egységnek (Például FMU-k.) a követelményeket vagy a köztük lévő logikai kapcsolatot.

Ezzel a megoldással, a fogalmak bevezetése abból áll, hogy minden elemet ellátunk a megfelelő annotációval és ezek alapján a leképezést létrehozó vagy felhasználó programok ezek segítségével tudják értelmezni, bejární a rendszermodellt szimulációs szemszögből nézve.

### 3.1.1.2. Szimulációs modellvetület bevezetése

A másik lehetőség, ha bevezetünk egy konkrét területet a modellben – egyfajta nézetet – amely leírja a szimulációs modellt. A rendszermodellezésben alapvetően jelen van ez a szemlélet, a rendszerek tervezésénél egy logikai és egy fizikai modellel dolgoznak a rendszermérnökök.

Ennél a megközelítésnél egy ezekhez hasonló, harmadik vetületet hozunk be a rendszermodellbe és a rendszermodell elemeit explicite összekötjük az azokat reprezentáló szimulációs elemek modelljével. Ezzel a megoldással a szimulációkhoz kapcsolódó fogalmak, csak a vonatkozó nézetben jelennek meg.

Magának a nézetnek a megvalósításához célszerű készíteni egy nyelvi kiterjesztést. Ez a SysML v2 adta lehetőség arra, hogy szakterület specifikus típusokat hozhassunk létre. Ennek az eszköznek a használata célszerűbb ennél a megközelítésnél, mert itt egy konkrét szakterület szerinti modellt alkotunk. Nincs szükség az annotációk flexibilitására, cserébe a használata egyszerűbb, jobban kikényszeríthető a megfelelő struktúra alkalmazása, mint ha csak különböző annotációkat adnánk az egyes elemekhez.

### 3.1.1.3. Lehetőségek összehasonlítása

Az annotáció alapú megközelítés előnye könnyen látható. Az, hogy a meglévő modell elemeit felcímkézzük, lehetővé téve a leképezést, azt jelenti, hogy minimális többletmunkára van szükség az eddigiekhez képest. A kevesebb nézet szerinti modellezés kevesebb munkát igényel, így a fejlesztés gyorsabban haladhat.

Továbbá mivel az annotációk bármilyen modellre elhelyezhetőek, annak struktúrájától függetlenül, így a már meglévő modellek esetében is könnyű bevezetni a szimulációkat a további fejlesztések támogatására. Ez a szempont jelenleg még kevésbé jelentős, mivel magának a SysML v2-nek a bevezetése sem kezdődött még el.

Ebből viszont az is következik, hogy más megközelítés használata esetén ez a legmegfelelőbb időpont, a szimulációk integrációjára is, mert jelenleg mindenképpen várható a modellek és módszertanok leváltása a SysML v2 szabvány megjelenése után. Egyszerűbb ha a többi komoly változás is megtörténik ezekkel együtt, mint utólag bevezetni új megközelítéseket amikor már végre újra normalizálódtak a fejlesztési folyamatok egy nagy váltás után.

Az annotációkon alapuló módszernek azonban van négy komoly hátránya is, melyek a második módszer használata mellett nem okoznak problémát.

Ha a teljes modellben jelen vannak az új szakterület fogalmai, az azt jelenti, hogy mindenkinek tisztában kell lenni azokkal, hogy dolgozni tudjon. Ez azt jelenti, hogy minden rendszermérnököt külön szimulációs technológiákra vonatkozó képzésben kell részesíteni. Ráadásul az extra képzésben részesített szakemberek nyilvánvalóan nem tudják olyan hatékonyan végezni ezt a munkát, mint azok, akik külön erre a területre specializálódtak.

Az annotációs módszer második hátránya, hogy bármilyen modellstruktúrát megenged, nem kényszeríti ki egy jól kidolgozott módszertan követését. Egy általános modellstruktúra leképezéséhez a feladatot ellátó szoftvereknek is jóval komplexebbnek kell lenniük, mintha egy előre ismert struktúra mentén dolgoznának.

A harmadik probléma, hogy az elemeken elhelyezett annotációk egyféle képpen címkézik fel a modellt, azaz egyetlen szimulációs modellt írnak le. A szimulációk során azonban rendszerint számos különböző modellt használnak, az aktuális igényeknek megfelelően, ezek között szükséges, hogy lehessen váltani, a komponensek különböző modelljeiből több összeállítást készíteni.

Ehhez vegyünk például egy robotot és annak tápegységét. Amikor a tápegységet fejlesztjük, fontos, hogy a benne zajló jelenségeket minél pontosabban modellezzük. Például egy kapcsoló üzemi tápegységnél a benne használt félvezető alkatrész pontos viselkedése

elengedhetetlen a hatékonyság és a kimenő vezetékeken megjelenő zaj optimalizálásához, a követelmények teljesülésének vizsgálatához. Azonban amikor a robotunk vezérlését teszteljük, akkor rendszerint érdektelen számunkra, hogy tökéletesen pontosan ismerjük a fenti paramétereket, az a fontos, hogy az adott mozgások okozta terhelést képes-e kielégíteni a rendszer, illetve körülbelül mekkora a fogyasztása. Ezt a problémát úgy oldják meg, hogy több, különböző részletességű modellt készítenek, az egyes feladatokhoz. Persze használható minden feladatra a legrészletesebb modell, azonban az extra részletek sokszor nem jelentenek lényeges különbséget, míg a szimuláció számításigényét jelentősen megemelik.

A szimulációs modellhez tartozó nézet bevezetése megoldja a fenti problémákat. Azaz, hogy egy konkrét modellrészletre korlátozzuk a speciális tudást igénylő munkafolyamatokat, lehetőség nyílik egyetlen, specializált szakemberekből álló csoportra bízni ezt a feladatot. A többi mérnöknek nem kell további képzés, a specializált szakemberek pedig jobb minőségű munkát végezhetnek a terület behatóbb ismeretének köszönhetően.

Az előre ismert struktúra és a doménspecifikus modellezési eszközök bevezetése megkönnyíti a modelleket és az eszköztámogatást készítő munkáját is.

Azzal, hogy a különböző szimulációs funkciókat nem kötjük hozzá direkt módon a rendszermodell elemeihez, lehetőség nyílik, hogy a szimulációs nézetben több szimulációs modellt is kapcsoljunk egy rendszerkomponenshez. Ez lehetővé teszi az eltérő részletességű modellek bevezetését, valamint a HiL és SiL technológiákhoz kapcsolódó szimulációs modellek is könnyen kezelhetők egy-egy extra verzióként.

Azonban legfontosabb különbség a két módszer között a fejlesztés nyomonkövethetősége. A szimulált rendszer egy virtuális reprezentációja a valósnak. A kritikus rendszereknél – ahol a rendszertervezésnek kiemelt jelentősége van –, mindennél fontosabb, hogy minden döntés, követelmény és felelősség nyomonkövethető legyen a teljes fejlesztési folyamatban. Azzal, hogy a rendszermodellt egyszerűen felcímkézzük a megfelelő jelölésekkel, elvesztjük az utolsó lépést az eddig gondosan épített nyomonkövethetőségből. Persze a konkrét elemekig ennek nyomán jutottunk el, így jelen van bizonyos mértékű nyomonkövethetőség, de ez messze nem tökéletes. Ez az alfejezet éppen arról szól, hogy a szimulációs modell kapcsán felmerülő legfontosabb tervezői döntéseket mérlegeljük, és éppen ezek azok, amelyeket annotációkkal nem tudunk nyomonkövetni, miközben a rendszertervezés legalapvetőbb szabálya, hogy minden lépés nyomonkövethető és indokolható kell legyen. Az új nézet bevezetése mellett nem áll fent ez a probléma, a rendszer vetületeiben eddig is minden tény szépen nyomonkövethető volt a teljes modellen keresztül.

A fentiek alapján látható, hogy az új nézet bevezetése jobban illeszkedik a technológiák alkalmazásához, jobb minőségű modelleket tesz lehetővé és a speciális szaktudás igényének csökkentésével az ilyen módszerek egyszerűbben és olcsóbban vezethetők be a gyakorlatban. Ezek alapján a további munka során erre a megközelítésre összpontosítunk.

### **3.1.2. Szimulációs modell architektúrák**

A szimulációs modellek esetében a változások azért szükségesek, hogy a rendszermodell megfelelően leképezhesse a szimulációs modelleket. Itt inkább tervezési konvenciókról, megkötésekről van szó, mivel egy teljesen szabadon készült modell leképezése jóval bonyolultabb feladat, mintha valamilyen előre ismert sémát használunk.

#### **3.1.2.1. Leképezés a Modelica teljes eszközkészletével**

A Modelica nyelvet kifejezetten fizikai rendszerek leírására és szimulációjára fejlesztették ki. Ennek köszönhető gazdag eszközkészlete van a különböző jelenségek leírására.

Az első megközelítés, hogy ezeknek az eszközöknek mindegyikét szabadon használjuk a rendszer leírásához, mivel a minél pontosabb modellek érdekében hozták létre ezeket

a nyelvi elemeket. Ilyen eszköz a külső változók használata, amelyek a környezeti hatások, mint például a gravitáció megadását teszik lehetővé. Ez azért hasznos, mert például égitestek modelljében, csak azt tüntetjük fel, hogy hat a testre ilyen erő és ez hogyan befolyásolja, azonban a számítását a környezetre bízza. Így könnyebben újrahasználható és logikusabban strukturált modelleket kapunk, hiszen nem egy adott égitesttől függ, hogy milyen erők hatnak rá hanem a környezetét alkotó égitestek alkotta rendszer sajátja ez a dinamika.

A másik fontos lehetőség az úgynevezett akauzális csatlakozók használata. Ezek irányítatlan kapcsolatokat jelölnek, ami azt jelöli, hogy a kapcsolat két vége valamilyen szempontból egyenlő egymással. Jó példa erre az áramkörti elemek beépített modellkönyvtára. A kapcsolat itt a kapcsolási rajzok vezetékeivel analóg módon viselkedik, az áram mindkét irányba folyhat az adott kapcsolaton és mindkét oldalán található komponensek befolyásolják ennek mértékét. Ez szintén olyan jelenség, amely a világban igen gyakran megfigyelhető, és rengeteg segítséget nyújt, amikor ilyen viselkedést kell modellezni.

Könnyen belátható, hogy ezeknek a lehetőségeknek a korlátlan kihasználása mennyivel egyszerűbbé teszi a szimulációs modellek megalkotóinak munkáját.

### **3.1.2.2. FMI kompatibilis leképezés**

Az FMI szabványt azért hozták létre, hogy az egyes modelleket könnyen át lehessen vinni egyik eszközről a másikba, illetve megoszthatóak, újrahasználhatóak legyenek az önálló logikai egységek. Ez egy olyan képesség, amit célszerű kihasználni az összetett rendszerek tervezése során, ez azonban megszorításokat kényszerít a modellekre.

Az eszközfüggetlenség ugyanis azt jelenti, hogy a szabvány csak a létező legalapvetőbb adatkapcsolatokat teheti lehetővé, amelyek minden eszközben elérhetőek lesznek. Ez a gyakorlatban azt jelenti, hogy kizárólag egész és racionális számok valamint logikai értékek szerepelhetnek a csomag ki és beviteli csatlakozóin és ezek a kapcsolatok kizárólag egyirányúak lehetnek.

Az ilyen megkötések mellett modellezett komponensek alakíthatóak át FMU-vá, hogy hordozhatóvá tegyék őket. Az FMI szempontjából csak a komponens és a környezetének kapcsolata számít, tehát csak azoknak a jeleknek kell megfelelni ezeknek a kritériumoknak, amelyek áthaladnak a komponens határain.

### **3.1.2.3. Lehetőségek összehasonlítása**

A fentiek alapján látható, hogy ebben a kérdésben nincsen olyan határozott válasz, mint a rendszermodellek esetében. Ahhoz, hogy a modellek minél jobban legyenek rendezve és minél egyszerűbben tegyék lehetővé a rendszerek leírását, használnunk kell az erre szolgáló eszköztárat. Ezek az eszközök azonban kizárják a modellek hordozhatóságát, amely elengedhetetlen komplex – több szakterületet érintő – rendszerek esetében, vagy több független szereplő együttműködése esetén.

Az első lépés ennek a problémának a feloldására, hogy jól definiáljuk, mely komponenseket szeretnénk FMU-vá alakítani, mivel a modellezési eszközök csak ezek határain kerülnek megszorításra.

Ha továbbra is gondot jelentenek a megkötések, akkor célszerű két részre bontani az adott komponenset. Az első rész adja az FMU-t, amely tartalmaz minden olyan logikát, amelyet nem szeretnénk megosztani, és a megszorítások mellett is elfogadható nehézséggel kommunikál a komponens modelljének maradék részével. A második rész a minimális modellet tartalmazza, amely a megszorított kommunikációs módokat átalakítja egy eszköz specifikus, de könnyen használható interfésszé. Fontos, hogy ezek az adapterelemek minél egyszerűbbek legyenek, mivel ezeket minden támogatni kívánt eszközben külön el kell

készíteni, valamint ezekre nem terjed ki a források védelme, mivel nem alakíthatóak FMU-vá.

Hasonló technikával próbálták már feloldani az akauzális kapcsolatok hiányát, de a módszer még nem elég kiforrott és az alkalmazása csak automatizáció után lehet valós alternatíva a gyakorlatban [7, 4].

Ezen felül, felmerült már az FMI szabvány kiterjesztése is az akauzális kapcsolatok támogatására, azonban ez még csak elméleti lehetőségként létezik, így szabványos megoldás egyelőre nem várható.

## **3.2. Modellezési módszertan**

Mint ahogy az a 2.3.1 részben is szerepelt, a SysML v2-höz még nincs kiforrott tervezési módszertan, így az eredményeink tesztelésének érdekében ki kellett dolgozni egy sajátot, amelybe integrálhatóak a szimulációs lépések is. A következőkben bemutatásra kerül a kidolgozott módszertan, amellyel az esettanulmányban használt rendszermodellek is készültek.

### **3.2.1. A fejlesztés során felmerült problémák**

Természetesen a modellezési módszerek nem úgy jönnek létre, hogy néhány hétnyi gondolkodás után megszületik a végleges megoldás, amely a továbbiakban gond nélkül alkalmazható. A dolgozatban leírt eredmények kidolgozása során számtalan különböző modell készült, az aktuális módszer segítségével, majd az ezek elkészítése során szerzett tapasztalatokra építve finomítottam tovább a módszereimet a következő iteráció előtt. A következőkben bemutatom ennek a folyamatnak a fontosabb tapasztalatait, amelyeknek mindenképpen tükröződnieük kellett a dolgozat következő részeiben javasolt megoldásban.

#### **3.2.1.1. A fizikai és logikai modellek elválasztásának hiánya**

A rendszertervezés során tipikusan két nézet jelenik meg. A logikai modell írja le, hogy milyen adatokkal avagy alapanyagokkal és mit csinál a rendszer. A fizikai modell a rendszer felépítését adja meg a pontos alkatrészekkel és összeköttetésekkel. A fejlesztés során fontos, hogy a két modell függetlenül fejlődhessen, főleg a kezdeti szakaszokban, mivel a túl korai kölcsönhatásuk könnyen kedvezőtlen irányba viheti a rendszer fejlesztését. Célszerű ezeket a modelleket külön forrásfájlokba szervezni, hogy elkerüljük a keveredésüket. Ez a tervezők munkáját is megkönnyíti, mivel a két modellt tipikusan eltérő szakemberek fejlesztik, akiknek a kevert források csak felesleges komplikációkat okoznának.

#### **3.2.1.2. Komponensek kohéziója a tervezés első fázisaiban**

Bár a fentiekből látható, hogy a logikai és fizikai modellek elválasztása fontos, azonban éppen ilyen káros az is, ha az adott komponenst különböző nézetekből leíró modellek között semmilyen kapcsolat nincs, hiszen ez téves feltételezés.

A nézetek teljes függetlensége arra is lehetőséget biztosít, hogy a tervezők véletlenül két külön alkatrész logikai és fizikai modelljét gondolják összetartozónak.

Persze a kész rendszerek ellenőrzése elméletben feltárná ezeket a hibákat, de akkor már számos kész, hibás példányt kellene javítani a széria hibák felfedezése után. Azért hozunk létre fejlesztési módszertanokat, hogy ne ezek az ellenőrzések tárják fel a hibákat, hanem lehetőségük se legyen létrejönni. Ez nemcsak jóval olcsóbb hosszútávon, de jóval biztonságosabb is, mivel az elképzelés, hogy az ellenőrzés feltár minden hibát feltételezi, hogy az ellenőrző folyamat maga nem hibás, ami természetesen nem feltételezhető kritikus rendszerek esetében.

### 3.2.1.3. Modellstruktúra átláthatósága

A fent említett két nézetén kívül még számos dolog jelenik meg egy rendszermodellben. Célszerű olyan módszertant létrehozni, amely mindegyiknek meghatároz egy pontos helyet, ahol a modellben elhelyezkedik, hogy könnyen lehessen navigálni a források között és mindenki számára átlátható legyen. A legjobb, ha ezek nem csak szabályokként jelennek meg, hanem a módszer olyan vázat is biztosít a munkának, amely kikényszeríti a megfelelő felépítést<sup>1</sup>.

### 3.2.1.4. Hierarchikus tervezés támogatása

A műszaki tudományok minden területén bevett gyakorlat, hogy különböző hierarchiakat vezetünk be a tervezés során. Ennek célja, hogy megkönnyítsék a tervek áttekintését és csökkentsék a komplexitás mértékét, hogy a problémák lokálisan mindig kezelhető keretek közt maradjanak. Ezek alapján nyilvánvaló, hogy csak olyan tervezési módszer fogadható el, amely támogatja a hierarchikus tervezést.

### 3.2.1.5. Szimulációs modellek vetületei, variánsainak reprezentálása

Jelen dolgozat egyik kiemelt célja, hogy kapcsolatot teremtsen a szimulációs és a rendszermodellek között egy új tervezési módszertan kidolgozásával. Mivel a szimulációk a fejlesztési folyamat részei, így ez alapvetően a rendszermodellben jelenti új elemek megjelenését.

A szimulációs modelleknek tehát valamilyen formában meg kell jelenniük a rendszermodellekben. Ezeknek a lenyomatoknak a kapcsán fontos kiemelni egy lényeges különbséget a két modell típus között. Míg egy konkrét rendszerhez jellemzően egy rendszermodell tartozik, addig szimulációs modellből több is készülhet egy komponenshez, hogy különböző szempontokból vagy különböző részletességgel mutassák be azt.

Ha az a célunk, hogy ennek a technológiának a teljes potenciálját kihasználjuk, akkor elengedhetetlen támogatni több ilyen modell elkészítését.

### 3.2.1.6. A modellek újrahasználásának megkönnyítése

A tervezési folyamatok során gyakran kerülnek felhasználásra már kész, általános célú elemek. Ennek előnye, hogy a gyakran használt elemeket elég egyszer megtervezni és tesztelni, utána számos alkalommal újrahasználhatóak.

Ha lehetőség van egyszerűen felhasználni az ilyen modelleket a tervezés során az sokat gyorsít és egyszerűsít a folyamaton. Ennek érdekében olyan módszertanra van szükség ami egységes megoldást ad erre a problémára, mivel így a gyártók készíthetik el a saját termékük modelljeit, ahelyett, hogy minden megrendelőjük saját verziót kellene készítsen.

Tehát minél jobb támogatást nyújt egy módszertan a modellek újrafelhasználására, annál jobban támogatja annak használóit és annál valószínűbb, hogy a gyakorlatban is jól alkalmazható lesz.

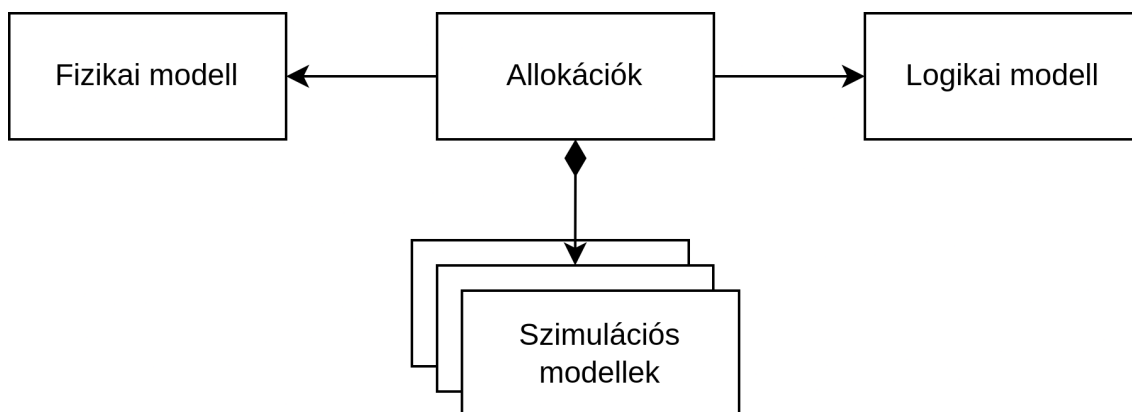
## 3.2.2. A kidolgozott módszertan javaslat bemutatása

A dolgozat fő eredménye egy a fenti elvárásoknak megfelelő módszertan javaslat elkészítése, melynek a végső verziója kerül most bemutatásra. A módszer evolúciójának részletes bemutatása túlmutat a dolgozat keretein, azonban a 3.2.1 részben kiemelt pontok alapján nyomon követhető az egyes elemek mögött meghúzódó motiváció.

<sup>1</sup>A gyakorlatban ez azt jelenti, hogy legyen kényelmes és egyszerű ezt használni a biztosított eszközökkel. A struktúrára tényleges garancia nehezen kényszeríthető ki, de a legkisebb ellenállás elve a gyakorlatban mégis eléri ezt az eredményt.

### 3.2.2.1. Alapgondolatok

A javasolt módszertan megértésének legjobb módja, az általa létrehozandó modellek szerkezetének megértése. Mielőtt a teljes modellt megvizsgálánk, vegyünk egy egyszerűsített képet az alapvető mögöttes elképzelésről, amelyre ráépülnek majd a különböző lépések. Ez az alapvető struktúra a 3.1 ábrán látható. A modelleknek ebben a felosztásban négy fő eleme van. A fizikai és logikai nézetek a 3.2.1.1 résznek megfelelően megjelennek, míg a kapcsolatukat az allokációk határozzák meg. Ez egyetlen rész, amely az eddigiek alapján meglepő, az a szimulációs modellek allokációkon belüli elhelyezése.



3.1. ábra. A modellek alapvető struktúrája

Ennek oka, az a mögöttes gondolat, hogy a rendszereket igazából egy allokációval lehet leírni. Mivel ez a gondolat elsőre nehezen átlátható vegyünk egy példát. Amikor egy autót akarunk leírni, logikusnak tűnik egy *Part*ként tekinteni rá – ez a jelenleg bevett módszer is –, mivel az 2.4.2 résznek megfelelően ez egy önmagában értelmes elem a modellben. Azonban amikor egy autóról beszélünk, akkor igazából valamilyen konkrét funkcióra és annak bizonyos megkötések mellett történő megvalósítására gondolunk.

Nem nevezzük valamit autónak, csak mert úgy néz ki, mint egy autó, vagy ha más funkciót lát el. Ha egy óriáskerék kabinjait autókból építjük meg, akkor azokat a továbbiakban nem tekintjük autónak, még akkor sem ha a folyamat során teljesen működőképesek maradnak, mert már nem az a funkció tartozik hozzájuk amit egy autótól elvárunk. Ugyancsak nem nevezzük autónak a Csillagok háborújának ikonikus homoksiklóját sem, hiába nyilvánvaló a nézőknek, hogy az autók megfelelői az adott világban. A szerkezete, amivel ezt megvalósítja eltér attól amit egy autótól elvárunk, mert forgó kerekek helyett lebeg és egy hajtómű tolja előre.

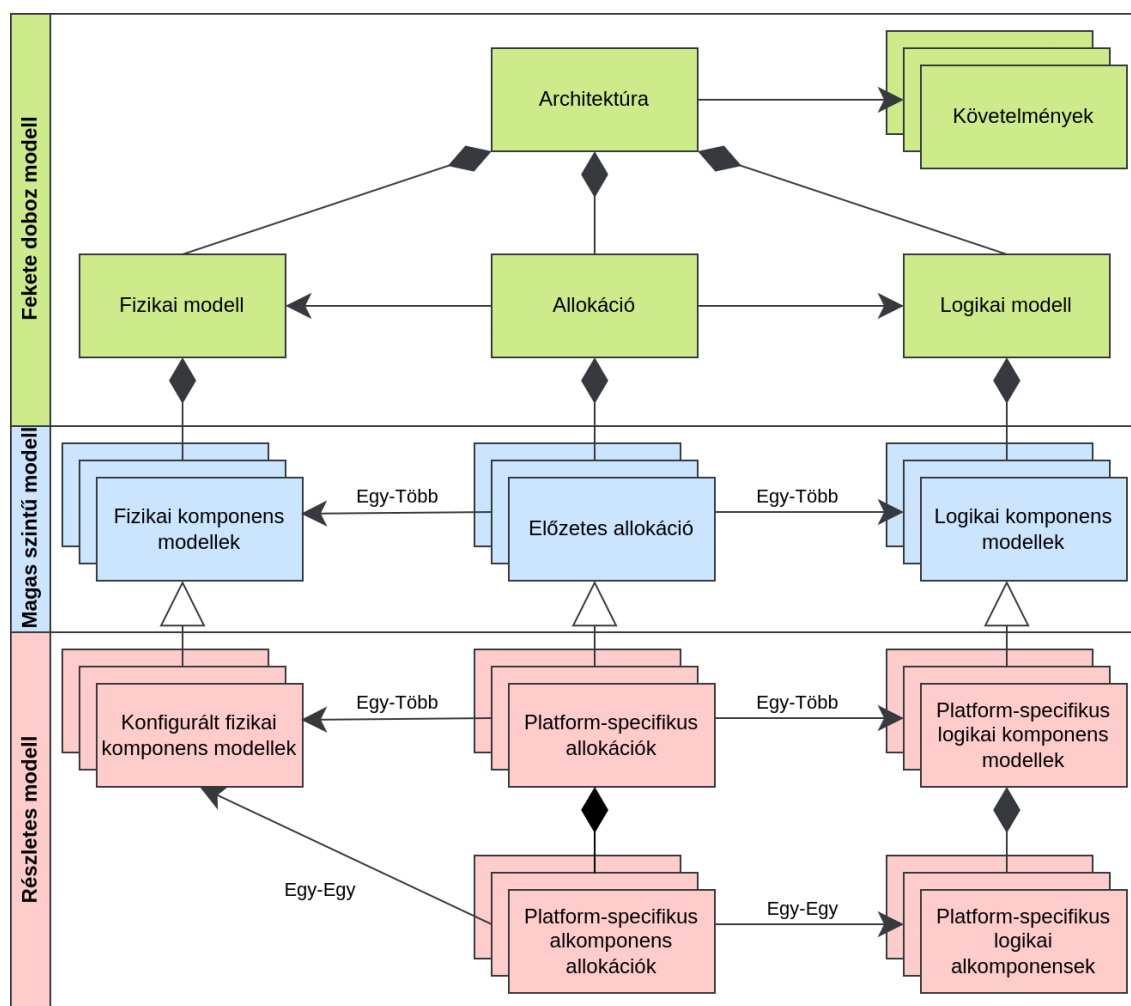
Elsőre nehéz látni, miért fontos, hogy magát a rendszert egy másik absztrakt elemmel írjuk le, ha az eddigi megközelítés is működött. Ennek oka a későbbiekben fog látszani, mivel ez a szemléletváltás tette kiemelt elemmé az allokációkat a kidolgozott módszerekben. Tervezés közben magától értetődő gondolat, hogy a modellek finomítása attól függ milyen funkciót és milyen eszközzel akarunk megvalósítani. Ennek ellenére a bevett tervezői gyakorlatban az allokációkat inkább befejező lépésnek tekintik, nem a folyamat mozgatórugójának. Persze így is létrejönnek végül a modellek, de csak több iteráció után, amikor sikerült a logikai és fizikai nézeteket egymáshoz igazítani. A szemléletváltás után, amikor kiemelt figyelmet szentelünk ennek a megfeleltetési folyamatnak, felfedezhető, hogy ezek az információk, amelyek a fent leírt összeillesztést vezérlik, végig jelen voltak az allokációkban, csupán figyelmen kívül maradtak a művelet helytelen szemlélete miatt.

A szimulációs modellek is ezért kaptak helyet, az allokációkon belül, mivel azok a konkrét rendszert reprezentálják, ami a két modellnézet közti megfeleltetés. Ha másképpen hoznánk létre ezt a megfeleltetést a nézetek között vagy bármelyiket módosítanánk,

akkor megváltoznának a szimulációs modellek is. Ebből látható, hogy ezek a modellek tényleg a rendszerhez, azaz az allokációkhoz tartoznak. Azzal, hogy az allokációk részei a szimulációs modellek egyértelművé válik melyik rendszerhez tartoznak, mert nem lehet őket elválasztani attól.

### 3.2.2.2. A modellek belső szerkezete

Maga a módszertan javaslat három nagyobb fázisra osztható, melyek a 3.2 ábrán láthatóak, különböző színekkel jelölve. Ezek egymás után, mindig az előzőre épülve jelennek meg, így a bemutatásuk is így történik a továbbiakban. Az ábrák konzisztensen ugyanazzal a színnel jelölik az adott fázishoz tartozó elemeket, megkönnyítendő az értelmezést.



3.2. ábra. A módszertan szerint készült modellek általános szerkezete

Az első fázis a *Fekete doboz modell* összeállítása. Ebben a lépésben még semmilyen implementációval kapcsolatos kérdésről nem szabad dönteni, a hangsúly a rendszerrel szemben támasztott követelmények meghatározásán van. Az implementációs kérdések mellőzése azért kiemelten fontos, mert ha a tervezők elkezdenek egy konkrét megoldásban gondolkodni akkor az túl korán beszűkíti a lehetőségeket a megvalósítások tekintetében és elterelheti a figyelmet fontos kérdésekről a követelmények terén. A követelményeken kívül olyan előkészítő lépések kapnak még helyet ebben a fázisban mint a projektváz kialakítása, esetleg alkatrészkönyvtárak összegyűjtése, bár itt vigyázni kell nehogy implementációs

kérdésekbe bonyolódjunk. A modell váza a 3.2 ábrán látható a zöld színnel jelölt területen amely ennek a fázisnak felel meg.

A *Követelmények*-ben találhatóak a rendszerrel által teljesítendő követelmények és kapcsolódó használati esetek, míg a többi elem a projekt vázát adja meg. Ebben a vázban megfigyelhető egy gyökérelem, amely összefogja a rendszert, valamint látszik hogy ennek a rendszernek, mint egésznek kell teljesíteni a követelményeket. A rendszer váza tartalmazza továbbá a 3.2.2.1 részben tárgyalt alapstruktúrát. Ezekben az elemekben találhatóak a rendszernek a környezete számára is ismert részei. Ezek az elemek a rendszer kontextusából, azaz beágyazó környezetéből ismertek. Például egy műhold esetében tudjuk, hogy milyen környezeti hatásoknak lesz kitéve a rendszer, milyen attribútumai vannak amelyek valamilyen érdekelt félnek fontosak, mint a tömeg és a méretek, illetve, hogy szükség lesz valamilyen vezeték nélküli kommunikációra amivel az eszköz kommunikálhat a földi irányítóközponttal. Utóbbinál fontos hangsúlyozni az implementációs kérdések mellőzését. Nem szabad belemenni olyan részletekbe, hogy rádiós, lézeres vagy valamilyen egyéb technológiát használunk majd, de az meg kell jelenjen, hogy lesz ilyen lehetőség, illetve a választandó technológia által teljesítendő követelményeket is össze kell gyűjtenünk. Ezen túl ennek a váznak a szerepe, hogy a továbbiakban erre tudnak majd ráépülni a különböző modellrétegek.

Ezután a *Magas szintű modell* elkészítése következik. A logikai modellben definiáljuk, hogy milyen részfeladatokra bontva és hogyan oldjuk meg a rendszertől elvárt működés biztosítását. A fizikai modellben megadjuk milyen alrendszerek és alkatrészek segítségével biztosítjuk ezeknek a logikai feladatoknak az elvégzését. Végül az *Előzetes allokációkkal* hozzárendeljük a feladatokat ezekhez a fizikai komponensekhez.

A javasolt módszertan szempontjából legfontosabb részlet, hogy ezek az *Előzetes allokációk* egyetlen logikai elemet több fizikaihoz is hozzárendelnek. Ez a végleges modellben már nem lenne előnyös, mert mindig pontosan egy felelősre van szükség egy adott feladat végrehajtásánál, de a magas szintű terveknel éppen ezt szeretnénk elérni. Ennek oka, hogy a fizikai és logikai nézeteket nem véletlenül választottuk külön, az egyes vetületekben a komponensek határai tipikusan eltérnek, átlapolódnak. Ha ez nem így lenne, akkor valószínűleg nem sikerült rendesen elválasztani a két nézetet, tehát ez egyik túl nagy mértékben meghatározta a másikat vagy már túlzottan belemerültünk az implementáció részleteibe. Mindkét eset könnyen vezethet szuboptimális megoldásokhoz és nehezen átlátható modellekhez. (Lásd: 3.2.1.1)

A célunk, hogy ezeket a többszörös allokációkat és mögöttes jelentésüket fel tudjuk használni a modell további finomítása során. Ez a rejtett információ az, amely korábban sokszor iteratív finomítással került feltárásra, pedig helyes allokációkkal és azok vizsgálatával könnyen átlátható. A következő fázis előtt fontos kitérni arra, hogy mi is okozza pontosan a többszörös allokációkat, illetve milyen információt hordoznak a későbbi lépések számára.

A fő ok, hogy a kérdéses logikai feladatot több fizikai komponens együttes munkája képes ellátni. Ebben az esetben ezt a feladatot tovább kell bontani az alapján, hogy milyen részfeladatokat melyik komponens valósít majd meg. Azért fontos, hogy ez a lebontás ne a magas szintű modellben történjen, mert annak feladata az adott probléma minél pontosabb megértése és hatékony megoldása, nem a konkrét implementáció elkészítése. Ha már ezen a szinten elvégeznénk a feladatok felosztását a fizikai komponensek függvényében, akkor az megnehezítené a modell átlátását, mert megfosztana minket a hierarchikus tervezési módszerektől amelyeket a 3.2.1.4 részben tárgyaltunk.

A másik lehetséges ok a biztonságkritikus rendszerekkel szembeni követelményekből és azok teljesítéséből adódik. Név szerint a redundáns alrendszerekből. Ha csökkenteni akarjuk egy alrendszer meghibásodásának esélyeit akkor gyakran az egyetlen ésszerű megoldás az, ha több példányt is beépítünk belőle a rendszerbe, amelyek külön-külön is képesek

ellátni a feladatukat. Ennek előnye, hogy annak jóval kisebb az esélye, hogy több is meghibásodik a kérdéses rendszerből<sup>2</sup>, így tovább növelhettük a rendszer megbízhatóságát. A redundancia implementálása szintén későbbi lépésekben célszerű, felesleges bizonyos részfeladatok duplikálásával bonyolítani az áttekinthetőséget célzó magas szintű terveket, amikor ennek oka a konkrét implementációval szemben támasztott elvárások.

A harmadik fázis a *Részletes modell* megalkotása. Ebben a lépésben finomítjuk tovább a fentiekben leírtak szerint a modellt, hogy megkapjuk a már említett egy-egy kapcsolatokat létrehozó allokációkat. A 3.2 ábrán látható, hogy ez esetben az új elemeket nem tartalmazza az eggyel korábbi lépés, hanem az új elemek *specializálják* a magas szintű komponensmodelleket.

Ennek oka, hogy a feladatoknak az egyes alkatrészek között több lehetséges felosztás is lehetséges. Azzal, hogy egy specializációval leválasztjuk ezt a réteget a magas szintű tervekről, lehetőségünk van több leszármazottat definiálni, amelyek a szimulációs modellek segítségével összehasonlíthatóak lesznek teljesítmény szempontjából. A több lebontás kapcsán gondolhatunk arra, hogy hogyan osztunk meg szoftverkomponenseket több processzor között a fogyasztás, válaszidő vagy pontosság alapján optimalizálva. De ugyanígy lehet szó arról, hogy a magas szintű modellben csak azt határoztuk meg, hogy egy adott gyártó melyik termékcsaládjából választunk például villanymotort egy adott feladatra, amely a vezérlés szempontjából már elegendő magas szinten, és csak a specializáció után választunk konkrét típust, amikor már pontosabb adataink vannak a várható terhelésről.

Maga a lebontás először a logikai elemek dekompozícióját jelenti a fizikai komponensek alapján, ezzel létrehozva a *Platform-specifikus logikai komponens modelleket*. A következő lépés a fizikai komponensek konfigurálása. Ez a különböző paramétereik meghatározását jelenti úgy, hogy ezzel készek legyenek a megfelelő feladatok ellátására. Ez lehet akár egy szabályozó paramétereinek hangolása, a memória kiosztása egy mikroprocesszoron, de konfigurálható logikai áramkörök esetében a konkrét áramkörök megtervezése és a megfelelő hardver erőforrások hozzárendelése is. Végül elkészítjük a *Platform-specifikus allokációkat* is, amelyekben alacsonyabb szintű allokációkkal teremtjük meg a már említett egy-egy kapcsolatokat a *Platform-specifikus logikai komponens modellek* és a *Konfigurált fizikai komponens modellek* között.

### 3.2.2.3. A fejlesztés menete

Az előző fejezetben már látható volt, hogy milyen főbb lépésekre tagolódik a rendszertervezés a javasolt módszerek használatával, de a modell szerkezete csupán egy egyszerűsített képet ad erről. Ebben a részben ez a folyamat kerül részletesebb bemutatásra.

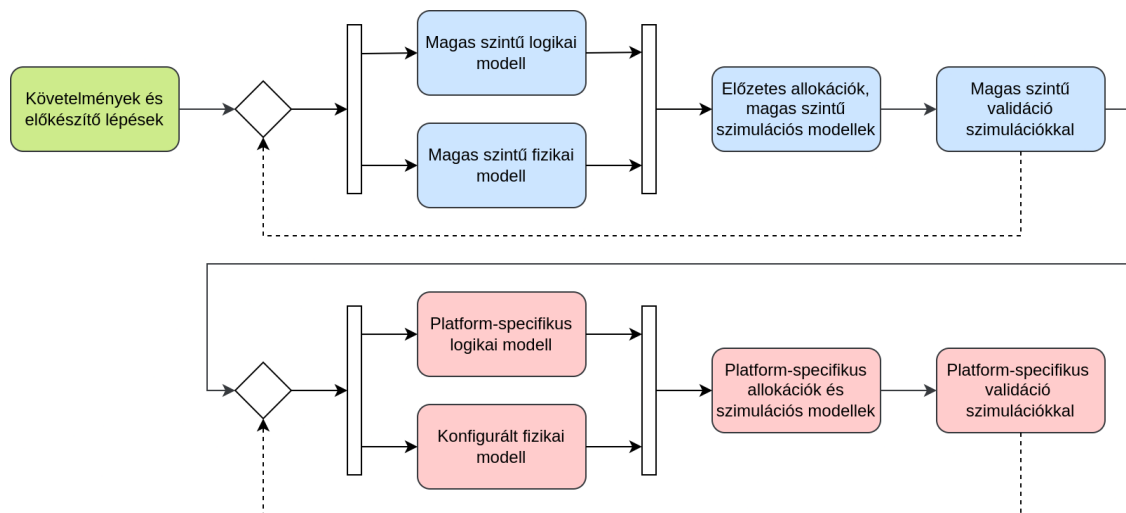
Maga a fejlesztési folyamat a 3.3 ábrán látható. Az ábra színezése megfelel a korábbiaknak elősegítendő a tájékozódást.

Az első fontos észrevétel a szimulációs modellek újbóli megjelenése. Míg a 3.1 ábrán látható, hogy az allokációk szimulációs modelleket tartalmaz(hat)nak, míg a 3.2 ábrán ezek nem voltak jelölve. Ez a változtatás csupán az ábra átláthatósága miatt történt, ezek a modellek továbbra is be vannak ágyazva az allokációkba, csupán az egyébként is összetett ábráról lettek lehvagyva a megértés megkönnyítésének érdekében.

Visszatérve a fejlesztés menetére, látható, hogy az első fázis a jelenlegi módszerek mellett egyetlen lépést igényel csak. A követelmények összegyűjtése és a javasolt módszertan által megkövetelt alapstruktúra létrehozása történik meg, a további munka előkészítésként.

---

<sup>2</sup>Ez az állítás csak független hibaokokat feltételezve érvényes. Ha ezt nem feltételezhetjük, akkor diverz tervezéssel védekezhetünk, ahol a duplikált alrendszerek nem pontos mások. Csak a velük szembeni követelmények azonosak, az implementáció minél eltérőbb, minimalizálendő az összefüggő hibaokokat. Például hiába használunk több GPS modult helymeghatározásra, ha nem az egyes modulok hanem maga a GPS hálózat válik működésképtelenné, akkor a redundancia önmagában nem nyújt védelmet.



**3.3. ábra.** A fejlesztési folyamat menete

A következő két fázis meglehetősen szimmetrikus, így azokat vizsgálhatjuk együtt. Először a logikai és fizikai modellek finomítása következik. Ez a két lépés haladhat párhuzamosan, mert az előző fázisokból szerzett információk alapján mindkét modell finomítása elkezdhető, de meg kell említeni, hogy a fizikai modell aktuális bővítésének befejezéséhez gyakran szükséges a logikai modell elkészítése. Erre jó példa a *Konfigurált fizikai komponensek* esetén egy mikrovezérlő memóriatartományának felosztása. Ugyanakkor a redundáns alrendszerek duplikálása nem igényli a pontos logikai modellt, így azt célszerű azzal párhuzamosan végezni.

A folyamatábránál az ilyen párhuzamosság jelölés azt jelenti, hogy a feladatok végrehajtásának sorrendje nem ismert, tehát nemcsak valódi párhuzamosságot jelöl a fenti ábra, csak pontos sorrend nélkül végezhető feladatokat. A kidolgozott módszertan javaslatban ez a sorrend az alkalmazott tervezési irányelvektől függ. Így megengedett top-down, bottom-up és kevert módszerek alkalmazása is. Fontos megemlíteni még az autóiparban egyre inkább elterjedő platformalapú módszereket, ahol a fizikai modell megelőzi a logikait, mert annak alapja többnyire már adott a rendszer tervezése előtt, lényegi változtatás csak az utolsó, konfigurációs fázisban történik rajta. Egy fázisokra bontott, és adott lépésben megengedőbb módszertan éppen azért célravezető, mert a lehető legkevesebb köti meg a módszer alkalmazóinak kezét az alkalmazandó tervezési irány tekintetében, elősegítve a széleskörű alkalmazhatóságát.

Ha elkészült a két modellnézet, akkor létrehozhatjuk az adott szinthez tartozó allokációkat és ezek alapján elkészítjük az új, részletesebb szimulációs modelleket. Végül mindkét fázisban felhasználjuk a szimulációs modelleket, hogy validálhassuk az új modelleket mielőtt továbblépnénk a következő fázisra. Természetesen nem csak szimulációt használunk validációra, de a javaslatok újdonsága ezeknek a technológiáknak a támogatása.

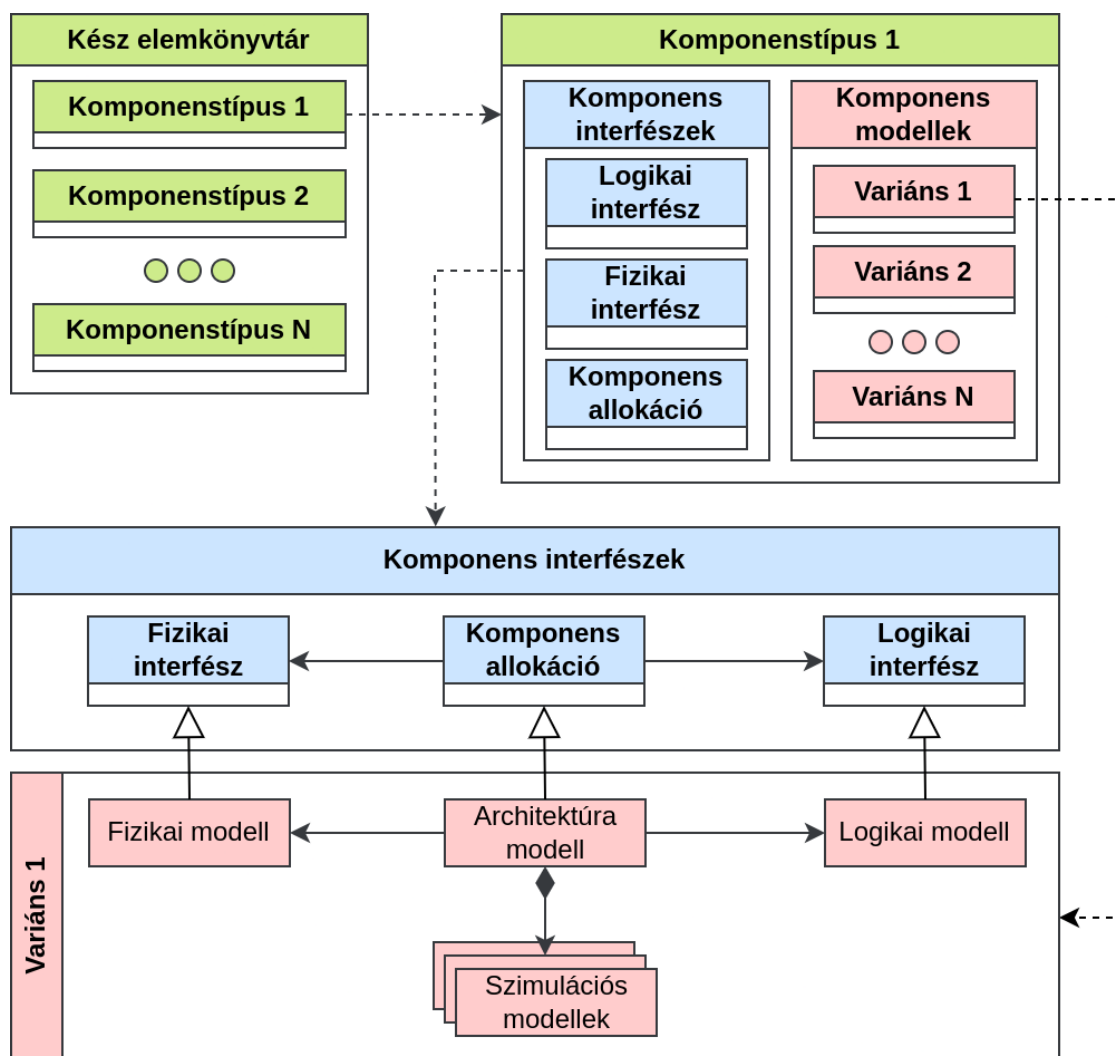
Amennyiben a validáció nem sikeres egy új iterációban megismételjük az adott fázist. Fontos, hogy ez az iteráció a validáció elbukásának következménye, a korábbi módszerekben megjelenő iteratív illesztést a logikai és fizikai modellek között már feloldottuk, ezt a típust pedig nem lehet. Ha sikeres volt a fázis végén a validáció, akkor tovább lépünk a módszertanban.

Ezeknél a validációs méréseknél van lehetőség végrehajtani a 3.2.2.2 részben említett kiértékelést a különböző implementációk között, mivel itt egyébként is futtatunk ilyen méréseket a továbblépés előtt. Ezen túl meg kell említeni, hogy a javasolt módszerek alkalmazhatóak az egyes alrendszerek kapcsán is. Az első fázis itt az adott alrendszerre

vonatkozó követelmények összegyűjtése, ezután pedig a módszertan lépései ugyanúgy ismételhetők az adott komponensen belül. Szintén alkalmazható ez a módszer a rendszer kontextusának modellezése során, ilyenkor gyakorlatilag egy olyan modellt alkotunk meg, amelynek a rendszerünk csak egyetlen eleme, célja pedig a környezeti elemek egymással és a rendszerrel való kölcsönhatásainak leírása. Ilyen modelleket minden módszertan használ a fejlesztés elején (Lásd: 2.5 rész), sajnos ennek a lépésnek a részletes vizsgálatára a dolgozat kereteiben nem volt lehetőség.

#### 3.2.2.4. A könyvtárak szerkezete és használata

Ahogy az a 3.2.1.6 részben is olvasható volt, a tervezés során gyakran felmerül az igény az egyes komponensek újrahasználására, esetleg külső beszállítótól való beszerzésére. Ennek megkönnyítésének érdekében a módszertannak támogatni kell komponenskönyvtárak létrehozását és használatát is. A módszertanhoz kidolgozott könyvtárstruktúra a 3.4 ábrán látható.



3.4. ábra. A komponenskönyvtárak szerkezete.

A könyvtáron belül komponenstípusok vannak. Ezekbe a kifelé – a paramétereiket leszámítva – azonosan megjelenő komponensek kerülnek csoportosításra, mint például szervomotorok, egyenáramú motorok, stb. Egy komponenstípuson belül definiálva van egy általános komponens-interfész, amely egy alapvető modell, amely a külvilág által látható

elemeket tartalmazza a komponenstípusra vonatkozóan. Ez az alapvető fizikai és logikai modellből valamint az azokat összekötő allokációs típusból áll.

Az allokációhoz azért célszerű külön típust bevezetni, mert megadható benne, hogy milyen típusú elemeket képes allokálni, így csökkentve a hibalehetőségeket.

Ezenkívül minden konkrét komponenshez tartozik egy-egy ilyen komponens-specifikus típus is, amely *specializálja* az általános verziót.

Ennek azon túl, hogy illeszkedik a kidolgozott módszertanba két előnye van. Először is a specializált allokációban szűkíthetjük a végekre vonatkozó típuskövetelményeket, hogy csak az összetartozókat fogadja el a konkrét komponensmodellekből, ezzel is csökkentve a modellezési hibák előfordulását. Másodszor pedig, az azonos típusú komponenseknek a legtöbb esetben csak apróbb eltérések jelentkeznek a szimulációs modelljeiben, a legjellemzőbb, hogy csak paramétereikben térnek el.

Ha a megfelelő szimulációs modellvázak szerepelnek az általános allokációban, akkor a konkrét típusnál elég a paramétereket felülírni és azt a néhány kisebb módosítást redefiniálni, ha ez szükséges.

Ezzel csökkenthető a kódDuplikáció, megkönnyítve a könyvtárak karbantartását. Ez azért is fontos, mert a 2.2.2.1-ben megismert FMI szabvány által definiált csomagok, amelyeket a gyártók biztosítani szoktak a megrendelőiknek szimulációs célra szintén parameterezhetők, illetve relative nagy méretűek a rendszermodellekhez képest. Ha tehát ezekből minél kevesebbet kell a könyvtárral együtt tárolni, akkor azok mérete radikálisan csökkenthető.

### 3.2.2.5. Szimulációs modellek használata

Mivel a módszertan kidolgozásánál külön cél volt a szimulációs technológiák integrálása, ezért tekintsük át az ezekre vonatkozó részeket.

A szimulációs modellek rendszermodellen belüli reprezentációival kapcsolatos elvárások a 3.2.1.5 részben olvashatóak. A legfontosabb követelmény, hogy több szimulációs modellt is támogasson a módszertan egy adott komponenshez, teljesül, mivel több is hozzárendelhető az allokációkhoz. Ezeknek a modelleknek a finomítása ráadásul folyamatosan történik a módszertan folyamán, a rendszerrel együtt fejlődnek és szerves részét képezik a fejlesztési folyamatnak, a tervezői döntéshozásnak.

Ezzel elérhető a 2.1.1 és 2.1.2 részekben leírt minden fejlesztési fázisban jelenlévő V&V tevékenység és gazdaságosság, mivel a módszer a lehető legkorábbi fázisban kiszűri a koncepcionális vagy implementációs hibákat.

Azzal, hogy a nagyobb fázisokban hierarchikusan helyezkednek el az egyes allokációk, támogatható, hogy egy magasabb szinten lévő modell hivatkozzon az általa tartalmazottakra, így a szimulációs modellekre is kiterjesztődnek a 3.2.1.4-ban tárgyalt hierarchikus módszerekre vonatkozó előnyök.

Mivel a hierarchiában minden szint több szimulációs modellt támogat, így hierarchikus szimulációs modellek használatával még egyszerűbbé válik a különböző részletességű szimulációs modellek készítése, mivel a az alkomponensek esetén elég a használt szimulációs modellek hivatkozásait redefiniálni.

## 4. fejezet

# Eredmények értékelése

A kidolgozott javaslatok esettanulmány alapján történő, illetve általános értékelése jelen fejezetben találhatóak.

### 4.1. Kvadkopter esettanulmány

A fent leírt módszerek tesztelése és továbbfejlesztése érdekében, szükség volt egy esettanulmányra, amelyen valószerű körülmények közt is ki lehet próbálni az elméleti eredményeket. Erre a célra egy olyan példa rendszerre van szükség, amely kellően bonyolult az előnyök kimutatására, az esetleges hiányosságok feltárására, de kezelhető méretű és lehetőleg széles körben érthető, hogy később demonstrációs eszközként szolgálhasson.

Ezek alapján a követelmények alapján egy kvadkopterre esett a választás. Pontosabban egy olyan négyrotoros drónra, amelynek az esettanulmányban a feladata, hogy szeles környezetben egy előre meghatározott pozícióban lebegjen.

Mivel az automatikus kódgenerálásra és a modellek konzisztenciájának ellenőrzésére még nincsenek eszközök, ezért ezeket a lépéseket manuálisan kellett elvégezni. A következőkben először a rendszermodell, majd a szimulációs modell kerül bemutatásra, az elkészítésük során gyűjtött tapasztalatokkal együtt.

#### 4.1.1. A rendszermodell

A rendszermodell esetében csak a főbb tervezési szempontok és a felépítés kerül bemutatásra, mert a dolgozat lényegének megértésében a konkrét forráskód elemzése nem segíti az olvasót. Ehelyett a struktúra bemutatása és a módszertan szempontjából lényeges részletek kiemelésére helyezem inkább a hangsúlyt ebben a részben.

A modellezett kvadkopter fizikailag igen egyszerű felépítést követ. Egy GPS és egy IMU modul segítségével méri a pozícióját és az orientációját. A példa egyszerűsége miatt ezek ideális szenzorként lettek modellezve. A módszertanban leírtak szerint ezt a modellt egy újabb iterációban lehetne tovább finomítani, de jelenleg tökéletesen megfelelő ez az absztrakció.

A szenzorok adatait egy mikrokontroller olvassa, amely futtatja a szabályozó algoritmusokat, majd az azok eredményeképpen adódó motorvezérlő jeleket továbbküldi a négy motornak.

A tervezés folyamán a motorokat kész komponensnek tekintettem amelyek a hozzájuk tartozó vezérlővel és rotorral egybeépítve készen kaphatóak. Ezek az egységek ipari drónok esetében valóban így kaphatóak katalógus szerint és saját adatlappal rendelkeznek, mert ezek a komponensek együtt határozzák meg a meghajtás különböző karakterisztikáit. Ezt azért fontos kiemelni, mert ez egy remek lehetőség kipróbálni a módszertanban javasolt

könyvtárak használatát, illetve később a szimulációs modellben az FMU-k bemutatására is remek lehetőséget nyújt.

A logikai modellben a szenzorok adatainak begyűjtése, a szabályozóalgoritmus és az az alapján történő beavatkozás jelenik meg magas szinten.

Utóbbi kettő azért érdekes, mert így a magasszintű modellben nem köteleződtünk el egy konkrét implementáció mellett, mivel számos megvalósítás lehetséges egy forgószárnyas drón megvalósítására. Ennek az előnye többek között, hogy a pozíciótartásért felelős szabályozó csak az összesített tolóerőt és a különböző forgástengelyek mentén jelentkező elmozdulást állítja így ez gyakorlatilag teljesen független a rotorok számától és elhelyezkedésétől. Ez persze azt jelenti, hogy a szabályozó kimenetét nem lehet azonnal a motoroknak küldeni, hanem előbb elő kell állítani belőlük a konkrét beavatkozásszervek utasításait.

Ezzel a megoldással a szabályozónk újrahasználmá válik elemmé, amelyet csak paraméterezni kell egy új hasonló rendszer esetében, ami egy valós projektnél előnyös gyakorlatnak számítana.

Ezek alapján a beavatkozó blokk a későbbiekben tovább lett finomítva, hogy az általános parancsok alapján meghatározza az egyes motorok tolóerejét az adott konfigurációhoz. Ezzel együtt finomításra került a két adatgyűjtési feladat a konkrét mérésre, valamint annak a szenzorból való kiolvasására és feldolgozására.

Ennél a két esetről jól megfigyelhető hogy a logikai és fizikai elemek átlapolódnak. Az adatgyűjtés két fizikai elemet is érintett, a beavatkozás pedig ötöt is, a transzformáció és a négy motor vezérlése által.

Ezek az átlapolódások jól láthatóak voltak a magasszintű allokációk elkészítésénél, a következő fázisban a felhasználással egyszerűen, egy lépésben elvégezhető volt a modellek finomítása az iteratív próbálkozás elkerülésével.

#### 4.1.2. A szimulációs modell

A rendszer fejlődésével készült egy a rendszert egyben reprezentáló szimulációs modell, amely a logikai és fizikai vetületet együtt reprezentálja az allokációnak megfelelően.

Ez alapján készült el később a Modelica modell is az eszköz beépített blokk-könyvtárával, amellyel grafikusán is megadhatóak a nyelv elemei.

Mivel a dolgozatban egyaránt foglalkoztam az FMU-kkal kompatibilis és a megkötések nélküli szimulációs modellekkel, ezért mindkét módszerrel készült egy-egy szimulációs modell. A megértést megkönnyítendő, a két modellben a különböző blokkokat eltérő színnel és betűjellel jelöltem az alábbiak szerint:

**Mikrokontroller** C(ontroller) narancssárga színnel

**Motor** M(otor) világoszöld színnel

**Váz** F(rame) világoskék színnel

**Szenzorok** S(ensors) lila színnel

A szerkesztő által használt jelölések a kapcsolatokhoz:

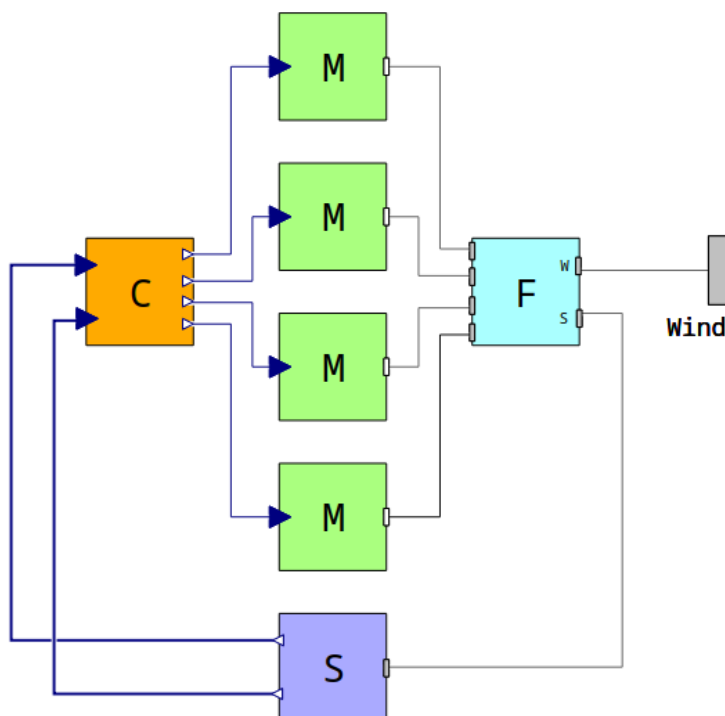
**Számadatok** Kék vonalak és háromszög alakú csatlakozók, valós számok átadására.

**Mechanika** Szürke vonalak és szögletes csatlakozók, az erők és forgatónyomatékok mindkét irányban terjednek rajtuk.

#### 4.1.2.1. Megkötések nélküli verzió

Ezt a modellt egyszerűen lehetett származtatni a rendszermodellből. Mivel a Modelica is lehetővé teszi a hierarchikus tervezést, ezért ezt a módszert alkalmaztam az átláthatóság érdekében. A hierarchia tetején a fizikai komponensek helyezkednek el, ezek belsejében vannak a logikai komponensek, amelyek a belső működést definiálják, illetve az adott alkatrész fizikai modellje is. Ezzel egy könnyen követhető modellstruktúrát kaptam, amely könnyen áttekinthető. A rendszernek ez a virtuális reprezentációja szerkezetileg alapvetően megegyezik a valósággal. Ez megfelel annak az elképzelésnek, hogy a szimulációs modell egy virtuális mása a valódi rendszernek.

A modell a fentiekben leírt jelöléseknek megfelelően a 4.1 ábrán látható. Jól megfigyelhető, hogy a motorok, szenzorok és a drón váza mechanikai kapcsolatban állnak egymással. Ugyanakkor a mikrokontroller a szenzorokkal és a motorokkal számadatokkal kommunikál. Természetesen a valóságban a mikrokontroller is a géphez lenne rögzítve, de ebben az egyszerűsített modellben a tömegét elhanyagoltam, az ábra áttekinthetősége érdekében.



4.1. ábra. A megkötések nélkül elkészített szimulációs modell.

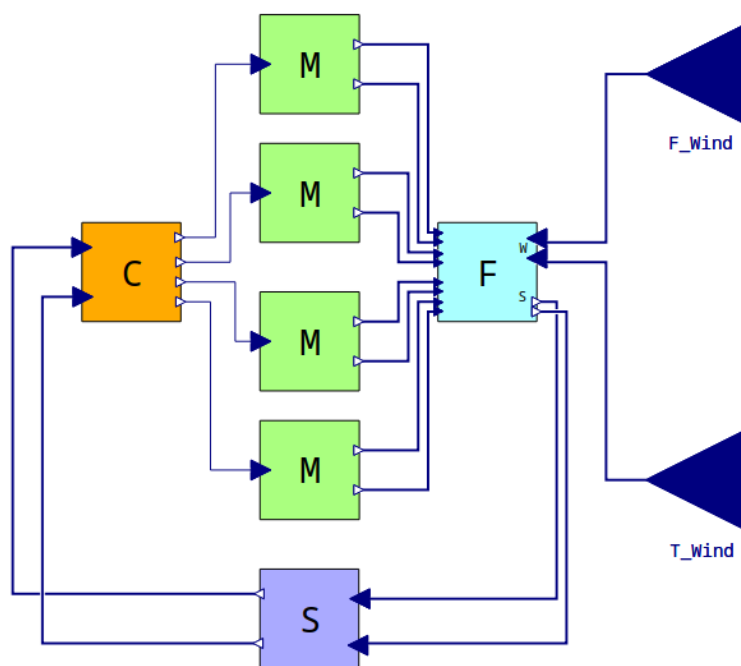
#### 4.1.2.2. FMI kompatibilis verzió

Ennek a modellnek a különlegessége, hogy míg az előzőben használható volt olyan kapcsolat, amely azt írta le, hogy a motorok fizikai kapcsolatban állnak a drón vázával, addig ebben a verzióban a komponensek határán csak logikai értékek valamint egész és racionális számok léphettek át. Ez persze feloldható volt a 3.1.2.2 részben leírt módszerrel, azonban ennek következtében a motorok és a váz között nem egy mechanikai kapcsolat volt, amin a megfelelő értékek minkét irányba terjedhettek, hanem az adott motor keltette tolóerő és forgatónyomaték, amely a váz modelljén belül került átalakításra fizikai értéké. Ennek következtében az, hogy minden motor mozgása kihat a teljes rendszerre, csak úgy volt modellezhető, hogy a mozgásmodell teljesen a vázban került megvalósításra.

Ennek kellemetlen következménye, hogy a motorok tömegét modellező blokkokat a vázban kellett elhelyezni, pedig ez egy külön alkatrész ami le is cserélhető egy másikra. Persze megoldható, hogy paraméterként adhassuk meg és az SSP (Lásd: 2.2.2.2 rész) szabvánnyal elméletben megoldható, hogy egy adott katalógusból választott motortípus kiválasztásával az egész modell átparaméterezhető legyen, de ez olyan komplex probléma ami nehezen automatizálható, sok hibalehetőséget tartalmaz és nehezen újrahasználható illetve karbantartható modellekhez vezet.

Az, hogy egy komponens modellje részben egy másikba kerül áthelyezésre, és nem a valós kapcsolatokat látjuk, hanem számértékek terjedését, nagyban megnehezíti annak áttekintését és felhasználását, de cserébe megoldható a modellek átadása anélkül, hogy a gyártó felfedné az implementáció részleteit, ami több ipari résztvevő között gyakran megjelenő igény a közös munka során. Mindemellett amíg a szabványt nem bővítik a szükséges akauzális kapcsolatok leírásához szükséges megoldásokkal, addig ez a technika csak nagyon indokolt esetben javasolt.

Az FMI kompatibilis modell a fentiekben leírt jelöléseknek megfelelően a 4.2 ábrán látható. Megfigyelhető, hogy az előző modellhez képest itt kizárólag számadatok formájában áramlik adat a komponensek között. A szenzorok esetében a mérést végző blokkok a vázba kerültek és a *Sensors* blokk csak a szenzorra jellemző zajt és egyéb paramétereket modellezi. A *Motor* blokkok pedig kizárólag az általuk létrehozott mechanikai hatások mértékét továbbítja a váznak amely a teljes dinamikai modellt implementálja, beleértve a határain kívül eső blokkok szerepét is.



4.2. ábra. Az FMI szabvány megkötései szerint elkészített szimulációs modell.

#### 4.1.2.3. Szimulációs eredmények

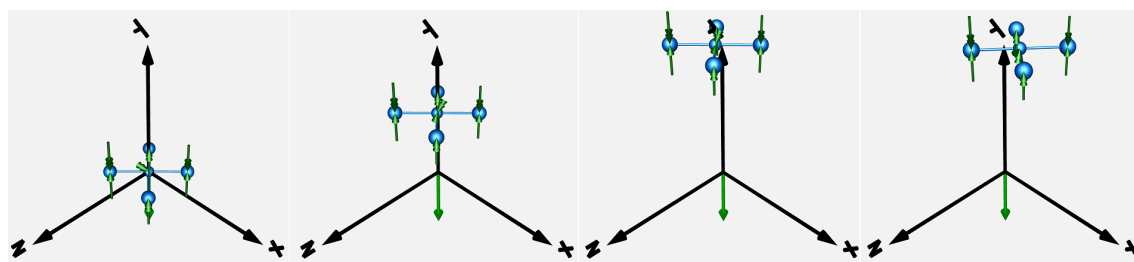
A szimulációkat futtatva a rendszer mindkét modell esetében felemelkedett, megközelítette az előre kijelölt pozíciót és azt a széllekedések mellett is tartotta elfogadható mértékű kitérések mellett.

A szimulációs eredményeket szemléltető animációról készült képek a 4.3 ábrán láthatóak. Az ábrán a fekete elemek a referencia koordináta-rendszert, a kékek magát a kvad-

koptert, míg a zöldek az erőket és forgatónyomatékokat szemléltetik. A szélső, motorokat jelképező gömbökre hatóerők és forgatónyomatékok a motorok által keltettek, amelyekkel beavatkozunk a rendszerbe. Az origóból induló lefelé mutató erő a gravitációt reprezentálja. A kvadrokopter testén ható erő és forgatónyomaték, amely az ábrán látható módon a többi nyíltól függetlenül mozog, a rendszerre ható szélzavarást reprezentálja.

A szimuláció paraméterei úgy lettek meghatározva, hogy a rendszer az origóból felszállva az  $y$  tengelyt jelképező nyíl hegyénél lebegjen.

Az ábrán látható, hogy a rendszer a szimuláció második felében -öt másodperc után- már elérte a kitűzött célt, a továbbiakban ezt próbálja tartani a szélzavarás ellenére, ez okozza a harmadik és negyedik képen látható kitéréseket. Ezeknek az eredményeknek az alapján, a tervezési folyamat és az esettanulmány is sikeresnek tekinthető.



4.3. ábra. A szimulációk eredménye pillanatképeken.

## 4.2. Értékelés a motivációk szempontjából

Először érdemes megvizsgálni, hogy a dolgozat motivációjául szolgáló szempontok szerint mennyire eredményes az eddigi munka.

### 4.2.1. Rendszertervezési szempontok

A modellalapú technikák általános előnyei közé tartozik a nyomonkövethetőség, a modellek konzisztenciája és a nézetek és dokumentumok automatikus generálhatósága. Mivel a dolgozatban javasolt módszertan is ilyen, modellalapú megközelítést alkalmaz, ezért ezeket nyelvi szinten teljesíti, illetve kompatibilis az eszközökkel amik képesek teljesíteni a konzisztencia esetében. A következő szempontok azok, amelyekre csupán lehetőséget biztosít a technológia, de ezek megvalósítása már a módszertannak köszönhető.

**Validáció és Verifikáció (V&V) minden fejlesztési fázisban** A módszertan bemutatása során látható volt, hogy a fejlesztés folyamán több alkalommal is szerepelnek validációs lépések, melyeknek célja a hibák mielőbbi kiszűrése. Ez komoly előrelépés a hagyományos módszerekhez és több modellalapú módszertanhoz képest, amelyek nem, vagy csak a fejlesztés végén tartalmazznak ilyen elemeket. (Lásd: 2.5 rész) Jelen dolgozat a szimulációs technológiákkal kapcsolatos validációra koncentrált, de természetesen ezekkel együtt végrehajthatóak más V&V lépések is ugyanazokon a pontokon.

**Modellek újrahazsnálhatósága** A javasolt módszerek kitértek olyan könyvtárak használatára amelyek újrahazsnálhatóvá teszik az általános modellrészleteket. Ez a szempont jellemzően nem jelent meg az eddigi módszertanokban, bár természetesen nem is zárták ki ezek használatát. (Lásd: 2.5 rész) A javasolt megközelítés esetén kiemelő, hogy a szimulációs technológiákkal integrált könyvtárakat javasol. Ezeknek a könyvtárba rende-

zése összhangban áll az általános szimulációs törekvésekkel és még inkább elősegíti a két terület integrálását.

#### 4.2.2. Szimulációs szempontok

Mivel a módszertan lehetőséget nyújt a szimulációs technológiák alkalmazására, így magában foglalja azok előnyeit is. Meg kell említeni azonban, hogy bár a dolgozat kitért a HiL illetve SiL szimulációkkal való kompatibilitásra és elvi szinten támogatja azokat, ennek pontos megvalósításával nem volt lehetőség foglalkozni az eddigi munka során.

#### 4.2.3. A két technológia integrálása szerint

Látható, hogy a kidolgozott módszerek számos előnnyel rendelkeznek a két érintett terület kapcsán, azonban célja a kettő ötvözése, az alábbiakban egyesével értékelem az ennek kapcsán felmerült előrelépéseket és hiányosságokat.

**Szimulációs modellek automatikus generálása** Mivel a két terület integrálásához szükséges a rendszermodellben egy külön reprezentációt készíteni a szimulációs modellekhez, ezért felmerül az igény, hogy ezek alapján automatikusan állítsuk elő őket.

A módszertan kidolgozása során részletesen vizsgáltam ezt a lehetőséget és kidolgozásra kerültek ennek alapvető sémái. Ezek az eredmények azért nem szerepeltek a dolgozatban, mert először későbbi fejlesztésre lettek félretéve, később pedig rábukkantam a 2.2.3.1 részben bemutatott SysPhS szabványra, amely az elérhető leírások alapján illeszkedik a kidolgozott alapelvekhez, így a saját megoldás helyett a két technológia ötvözése lett a cél ezen a területen.

Az esettanulmány alatt, amikor a szimulációs modellreprezentációkat és a szimulációs modelleket is kézzel kellett elkészítenem, az eddigiek mellett felmerült, hogy magukat a modellreprezentációkat is lehetne részben automatikusan generálni, ezzel még jobban kiterjesztve ezeket az előnyöket.

**Munkafolyamatok konzisztenciájának garantálása** Az integrált technikák esetében ez a cél különböző modelltípusok konzisztenciáját jelenti. A módszertan javaslatot tett olyan modellstruktúrákra, amelyek megkönnyítik ennek ellenőrzését, illetve lehetőséget biztosít a kódgenerálásra, amely biztosítja ennek a konzisztenciának a felhasználóbarát megvalósítását is.

Ami a szimulációk eredményeit illeti, ezeknek a garantált konzisztenciájához szükséges, hogy a szimuláció eredménye automatikusan visszakerüljön a rendszermodellbe. A leképezés ezen irányára nem tért ki a módszertan, bár az esettanulmány tapasztalatai alapján ez módszertantól független eszközökkel biztosítható, ha a tervezési folyamatban van hely ilyen vizsgálatoknak. A 3.2.2.3 rész végén találhatóak javaslatok ilyen lépések elhelyezésére, így egy ilyen eszköz elméletileg gond nélkül használható a javasolt módszerekkel.

**Integrált V&V** A két előző pont teljesülése esetén megvalósuló integrált V&V folyamatok a fentiekben leírtak alapján elő lettek készítve, azonban tényleges megvalósításuk még várat magára, főleg mivel ehhez már szükség van az említett kétirányú kapcsolatra.

**Rendszer-identifikáció támogatása paraméter-visszavetítéssel** Ez a technológia ugyanarra a kétirányú kapcsolatra támaszkodik mint ami az előző két pontban is szerepelt, így ennek megvalósítása szintén külső eszközök fejlesztését igényli, de a tapasztalatok

alapján, ha elkészülnek ilyen keretrendszerek, akkor azok használhatóak lesznek a leírt módszerekkel.

### 4.3. Értékelés a felhasználási tapasztalatok szerint

Az esettanulmány során készített modellek fejlesztésénél a leírt módszerek hatékonynak és kényelmesnek bizonyultak.

A szimulációs modelltranszformáció alapvetően repetitív és monoton feladatnak bizonyultak, ahogy azt a 3.1.1.3 részben előrevetítettem, azonban ennek az elvégzése hosszútávon automatizálандó feladatként szerepelt a kezdetektől fogva. A tapasztalatok alapján ez a feladat valóban sematikus és jól automatizálható.

A finomítások során szükséges adatok könnyen kinyerhetőek a javasolt allokáció-centrikus modellekből, nemcsak a tervezők, de az automatizált eszközök számára is. Mindkét eset megkönnyíthető lenne viszont nyelvi kiterjesztések használatával, amely érdekes továbbhaladási irányt jelent.

## 5. fejezet

# Konklúzió és további munka

### 5.1. Összefoglalás

A dolgozatban leírt munka során először összegyűjtöttem a modellalapú rendszertervezés és a szimulációs technológiák külön-külön illetve együttes alkalmazásának előnyeit. Ezután megvizsgáltam egy, a fenti technikákat integráló SysML v2 alapú tervezési módszertan által teljesítendő követelményeket, javaslatot tettem több különböző implementációs irányra, majd kiértékeltem ezeket.

Az eredményeket felhasználva kidolgoztam ilyen módszertan alapjait és egy esettanulmányon keresztül demonstráltam az új, fentieknek megfelelő módszerek gyakorlati alkalmazását, ami számos az integrációból fakadó előnnyel jár, valamint alapot biztosít továbbiak elérésére, ha a hozzájuk szükséges fejlesztőeszközök létrejönnek.

A javasolt módszereknek három fontos eleme van, amelyek nem, vagy csak részlegesen jelennek meg a korábbi módszertanokban. A dolgozat legfontosabb új gondolata az allokációk köré épülő rendszerszemlélet, amely a javasolt módszerek alapkövét képezi és tényleg különlegesen számítt a korábbi módszertanokhoz képest. Ezután fontos megemlíteni a modell hierarchikus finomítását, amellyel jól elhatárolhatóak, a különböző absztrakciós szintjei a modellnek és kényelmesen teszi lehetővé különböző implementációs lehetőségek összehasonlítását. Ezt a megoldást SysML v2 új nyelvi elemeinek a felhasználása tette lehetővé. Végül, de nem utolsó sorban, a javasolt megközelítésnek szerves részét képezik a szimulációs technikák. Erre a feladatra számos korábbi módszertanban tettek kísérletet, de azokban az esetekben mindig problémát jelentett, hogy ezeket utólag próbálták egy addigra kiforrott koncepcióba beilleszteni.

A munkám során feltárássra került, hogy a javasolt előnyök némelyike külön eszközöket igényel, amelyek csak a fejlesztési folyamatban igényelnek helyet a használatukhoz, magától az alkalmazott módszertantól függetlenek. Ezek esetében megvizsgáltam, hogy használhatóak-e ezek a technikák a javasolt megközelítésekkel, melynek során nem találtam kizáró okot az együttes használatra.

### 5.2. További kutatási és fejlesztési lehetőségek

A rendszertervezési módszertanokban (Lásd: 2.5 rész) rendszerint a rendszer modellezése mellett nagy hangsúlyt helyeznek annak kontextusának, azaz beágyazó környezetének a modellezésére. Sajnos az eddigi munkám során ezt nem volt még lehetőségem kidolgozni, ezért szeretném először ezt a hiányosságát pótolni a javasolt módszertannak, hogy minél közelebb kerüljön egy teljes, gyakorlatban is jól alkalmazható módszertanhoz.

A kódgenerálás kapcsán felmerült, hogy ez az elem elengedhetetlen ahhoz, hogy egy módszertan megfelelően eredeti rendeltetésének, és bemutatásra került a fejlesztés alatt álló

SysPhS szabvány is, amely alkalmazásával ez a feladat gyorsabban megoldható, mint saját eszköz fejlesztésével. Ezek alapján konzulensemmel fel is vettük a kapcsolatot a szabványon dolgozó egyik csapattal és megkezdjük az együttműködést a két eszköz összekapcsolása érdekében.

Jelen dolgozat elsődlegesen a validációs technológiák közül a szimulációk integrálásával foglalkozott, mert ezek igényelnek külön környezetekben készített és futtatott modelleket, azonban a jövőben szeretnék több, más V&V technikát is integrálni a javasolt módszerekbe és ezek függvényében bővíteni a bemutatott esettanulmányt. Az esettanulmány kidolgozása folyamán az eddig kidolgozott módszerekben azonosítottam több általános mintát, melyeknek egy SysML v2 nyelvi kiterjesztésben való összegyűjtése tovább egyszerűsítene azok használatát, amely előnyös lenne egy teljes módszertan kidolgozása esetén. Természetesen a nyelvi kiterjesztés elkészültével újra frissíteném az esettanulmányt, hogy használható legyen az új, kiforrottabb módszerek bemutatására, valamint referenciamodelleknek.

# Irodalomjegyzék

- [1] autoEEsystemEngineer: OOSEM (Object-Oriented Systems Engineering Methodology). URL <https://www.autosyseng.com/oosem/>.
- [2] Manas Bajaj – Sanford Friedenthal – Ed Seidewitz: Systems modeling language (sysml v2) support for digital engineering. 25. évf. 1. sz., 19–24. p. ISSN 2156-4868.
- [3] Capella: Arcadia method at a glance!  
URL <https://mbse-capella.org/arcadia.html>.
- [4] Garron Fish: Using physical connectors with an fmu. URL <https://www.claytex.com/tech-blog/using-physical-connectors-with-an-fmu/>.
- [5] Daniele Gianni – Andrea D’Ambrogio – Andreas Tolk: *Modeling and Simulation-Based Systems Engineering Handbook*. Taylor & Francis Group, 513. p. ISBN 9781138748941.
- [6] Git: The advantages of git.  
URL <https://git-scm.com/about/branching-and-merging>.
- [7] Yutaka Hirano – Satoshi Shimada – Yoichi Teraoka – Osamu Seya – Yuji Ohsumi – Shintaroh Murakami – Tomohide Hirono – Takayuki Sekisue: Initiatives for acausal model connection using fmi in jsae (society of automotive engineers of japan). 118. évf., 795–801. p. ISSN 1650-3686.
- [8] Inquiry Board: ARIANE 5 Flight 501 Failure.  
URL <http://sunnyday.mit.edu/nasa-class/Ariane5-report.html>.
- [9] Intel Corporation: Intel® cofluent™ methodology for sysml.  
URL <https://www.intel.com/content/dam/www/public/us/en/documents/white-papers/cofluent-methodology-for-sysml-white-paper.pdf>.
- [10] LieberLieber GmbH.: Lemon tree.  
URL <https://www.lieberlieber.com/lemontree/en/product/>.
- [11] Junda Ma – Guoxin Wang – Jinzhi Lu – Hans Vangheluwe – Dimitris Kiritsis – Yan Yan: Systematic Literature Review of MBSE Tool-Chains. 12. évf. 7. sz., 3431. p. ISSN 2076-3417.
- [12] Azad M. Madni – Michael Sievers: Model-based systems engineering: Motivation, current status, and research opportunities. 21. évf. 3. sz., 172–190. p. ISSN 1520-6858.
- [13] Mathworks: Dynamical system modeling using neural ode.  
URL <https://www.mathworks.com/help/deeplearning/ug/dynamical-system-modeling-using-neural-ode.html>.

- [14] Mathworks: System Composer - Design, analyze, and simulate system and software architectures.  
URL <https://www.mathworks.com/products/system-composer.html>.
- [15] Mathworks Support Team: What are MIL, SIL, PIL, and HIL, and how do they integrate with the Model-Based Design approach? URL <https://www-ah.mathworks.com/matlabcentral/answers/440277-what-are-mil-sil-pil-and-hil-and-how-do-they-integrate-with-the-model-based>
- [16] Modelica Association: Functional Mock-up Interface Specification.  
URL <https://fmi-standard.org/docs/3.0.1/>.
- [17] Modelica Association: Modelica® – A Unified Object-Oriented Language for Systems Modeling. URL <https://specification.modelica.org/maint/3.6/MLS.pdf>.
- [18] Modelica Association: System Structure and Parameterization. URL <https://ssp-standard.org/publications/SSP101/SystemStructureAndParameterization101.pdf>.
- [19] NASA Safety Center: System failure case studies: Lost in translation.  
URL [https://sma.nasa.gov/docs/default-source/safety-messages/safetymessage-2009-08-01-themarsclimateorbitertermishap.pdf?sfvrsn=eaa1ef8\\_4](https://sma.nasa.gov/docs/default-source/safety-messages/safetymessage-2009-08-01-themarsclimateorbitertermishap.pdf?sfvrsn=eaa1ef8_4).
- [20] Object Management Group Inc.: SysML Extension for Physical Interaction and SignalFlow Simulation. URL <https://www.omg.org/spec/SysPhS/1.1/PDF>.
- [21] Timothy W. Simpson – Zahed Siddique – Jianxin Roger Jiao: Platform-based product family development. 1–15. p.
- [22] Speedgoat GmbH: Rapid Control Prototyping. URL <https://www.speedgoat.com/solutions/testing-workflows/rapid-control-prototyping>.
- [23] Tim Weilkiens: *SYSMOD - the systems modeling toolbox*. MBSE4U booklet series sorozat. 3rd edition, version 4.2. kiad. MBSE4U. ISBN 9783981852981.