



Budapesti Műszaki és Gazdaságtudományi Egyetem

Gépészmérnöki Kar

Gyártástudomány és -technológia Tanszék

Kardos Csaba

Gyártórendszerek modell-rekonstrukciója

TDK-dolgozat

KONZULENSEK:

Dr. Monostori László

Popovics Gergely

Budapest, 2011

Tartalomjegyzék

Rövidítésjegyzék	4
Kivonat	5
1. Programozható Logikai Vezérlők	6
1.1. Bevezetés: a PLC-k általános jellemzői	6
1.2. PLC-k hardver architektúrája	7
1.3. PLC-k szoftveres felépítése	8
1.3.1. PLC programok szerkezete	8
1.3.1.1. PLC-k programozási nyelvei	10
2. Ipari automatizálási rendszerek	11
2.1. Az ipari automatizálási rendszerek elemei	11
2.2. PLC-k szerepe az ipari automatizálási rendszerekben	12
2.3. Az ipari automatizálási rendszerek felépítésének fejlődési iránya	13
3. Szimuláció	15
3.1. Bevezetés	15
3.2. Gyártórendszerek szimulációja	15
3.3. Automatikus szimuláció-építés	16
4. A megvalósított feladat ismertetése	18
4.1. A tárgyrendszer bemutatása	18
4.2. Kódelemzés	20
4.2.1. Kódelemzés a ProGrammar környezet segítségével	20
4.2.2. A kódelemzés eredménye	21

4.3.	Az elemzett kód feldolgozása	23
4.3.1.	Kódfeldolgozás a Mathematica programmal	23
4.3.1.1.	Elemtípus keresés	25
4.3.1.2.	Redundancia kezelése	26
4.3.1.3.	Azonosítatlan elemek kezelése	26
4.3.1.4.	Elágazások kezelése	27
4.4.	Adatszerkezet a modellépítéshez	28
4.5.	Az létrehozott rendszer felépítése	29
5.	Összefoglalás, továbbfejlesztési lehetőségek és értékelés	31
5.1.	Összefoglalás és az eredmények értékelése	31
5.2.	Továbbfejlesztési lehetőségek	32

Rövidítésjegyzék

API Application Programming Interface.

DB Data Block.

DES Discrete Event Simulation.

EE Emelőasztal Elem.

FB Function Block.

FE Forgóasztal Elem.

FG Function Group.

GDL Grammar Definition Language.

GE Görgőspálya Elem.

IL Instruction List.

LE Lift Elem.

OB Organization Block.

PLC Programmable Logic Controller.

POU Program Organization Unit.

SZE Szíj Elem.

Kivonat

Napjaink gyártórendszereiben a magasabb minőség és gazdaságosabb termelés érdekében az automatizálás egyre nagyobb szerepet kap. Emiatt az automatizálási rendszerek egyre átfogóbb feladatokat látnak el és összetettebbé válnak [7]. Ezen tulajdonságok miatt az automatizálási rendszereket figyelembe kell venni a gyártórendszerrel kapcsolatos tervezési, elemzési és újrakonfigurálási feladatoknál is. Ezekre a feladatokra széles körben elterjedt a különböző szimulációs eszközök használata (pl.: Diszkrét idejű eseményvezérelt simuláció – Discrete Event Simulation (DES)) [17].

A tradicionális szimulációs és szimulációs-modellépítő eljárások azonban magasabb absztrakciós szinten reprezentálják a gyártórendszerek működését, ezért csak korlátozott lehetőségeket nyújtanak az automatizálási rendszerek alacsony szintű vezérléseit is tartalmazó szimuláció készítésére [16, 7]. Emiatt egy ilyen integrált szimulációs-modell létrehozásához a modellépítés során kapcsolatot kell teremteni a gyártórendszer eddig elkülönült szintjei között. A szimulációs-modellépítés azonban (főként az adatgyűjtés nehézségei miatt) idő- és erőforrásigényes feladat [21], amit viszont lényegesen hatékonyabbá tehet egy automatizált eljárás megalkotása [3, 18]. Egy ilyen eljárás egyik alapvető feladata az adatgyűjtés automatizálása [21], ami az alacsony szintű vezérlésekben található adatok felhasználásával valósítható meg. Ezen adatok hozzáférése és kezelése olyan új megoldások kidolgozását igénylik, melyek eltérnek a hagyományos szimulációs-modellépítés során alkalmazott eljárásoktól.

A TDK dolgozatomban bemutatott módszer segítségével a vezérlési programok és a hozzájuk kapcsolódó adatstruktúrák visszafejtésével létrehozható az automatikus szimulációs-modell építő rendszer működéséhez szükséges adatbázis. Ez az adatbázis képezi a vezérlés és a szimuláció közötti kapcsolati réteget, amin keresztül a DES-rendszer a szimulációs vizsgálathoz szükséges adatokkal ellátható.

1. fejezet

Programozható Logikai Vezérlők

Napjainkban a programozható logikai vezérlők (Programozható Logikai Vezérlő – Programmable Logic Controller (PLC)) a leggyakrabban alkalmazott eszközei az ipari folyamatirányításnak [1]. Ezért ebben a fejezetben röviden bemutatásra kerülnek a PLC-k működésével kapcsolatos alapvető ismeretek.

1.1. Bevezetés: a PLC-k általános jellemzői

A PLC-k megjelenését megelőzően jelenlévő vezérlő eszközök számos olyan nehézséggel küzdöttek, mely korlátozta az alkalmazhatóságukat (pl.: mozgó alkatrészek, nehezen módosítható struktúra). A mikroprocesszorok fejlődésével lehetőség nyílt egy, a korábbi technológiák hátrányait nélkülöző, eszköz megalkotására, így jött létre a PLC. A PLC alkalmazásának főbb előnyei a következők:

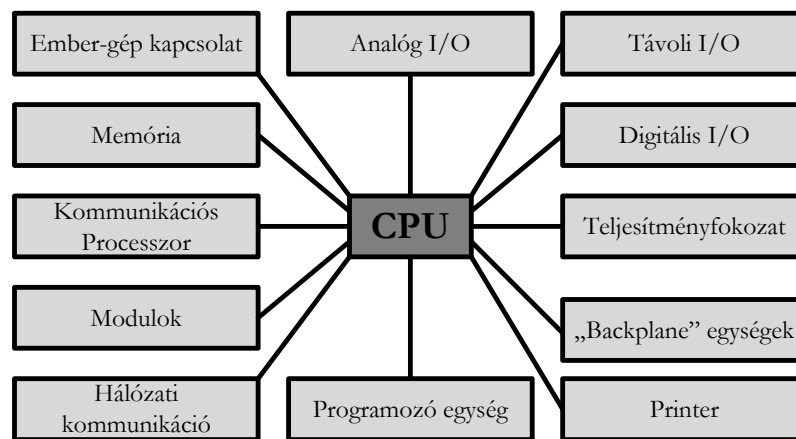
- Valós idejű működés
- Mozgó alkatrészek hiánya
- Univerzális hardver architektúra
- Nagyobb flexibilitás
- Programozás számítógép segítségével, grafikus felületen keresztül
- Fejlett kommunikációs képességek

A programozható vezérlőkkel kapcsolatos általános információkat tartalmazó IEC 61131-1 szabvány az alábbi meghatározását adja a PLC-nek: *„Ipari célú felhasználásra tervezett digitális működésű elektronikai rendszer, mely programozható memóriában tárolja a felhasználó-orientált logikai, aritmetikai, időzítési és számlálási utasításokat, hogy analóg vagy digitális*

bemeneteken és kimeneteken keresztül különböző gépeket irányítson. Továbbá a PLC és a hozzá kapcsolódó perifériák egyszerűen integrálhatóak egy ipari automatizálási rendszerbe.” [8]

1.2. PLC-k hardver architektúrája

A hardveres felépítés szempontjából vizsgálva a PLC-et, az IEC szabvány által adott definíció kiegészíthető: „a PLC egy olyan mikro-processzor alapú valós-idejű működésű számítógép, ami jelentős mennyiségű I/O és egyéb kommunikációs tevékenységet, bit-orientált és lebegőpontos számításokat végez” [14].



1.1. ábra. PLC funkcionális felépítése [14]

A hardver felépítés alapján kijelenthető, hogy a PLC valójában egy speciális számítógép, az általános felhasználási célú számítógépekkel összehasonlítva azonban több eltérés is megfigyelhető. Az eltérések oka elsősorban a különböző felhasználási területekben keresendő. A főbb különbségeket az 1.1 táblázat tartalmazza.

A számítógépekhez hasonlóan a legtöbb PLC is moduláris felépítésű, különböző bővítő egységek beépítésével tovább szélesíthető az alkalmazhatóság és növelhető a flexibilitás [11]. Megjegyzendő, hogy az általános felhasználású számítógépek fejlődésével egyre gyakoribbak a PC-alapú vezérlést megvalósító rendszerek. Különösen alacsony-költségű rendszerekben jellemző a használatuk [20]

PLC	Számítógép
Szélsőséges körülmények között is képes működni	Nem képes a szélsőséges körülmények közötti működésre
Nagy zavarvédeettségre optimalizált	Nagy működési sebességre optimalizált
Beépített parancsértelmező	Változatos programozási lehetőségek
Nem áll rendelkezésre másodlagos memória a PLC-n belül	Rendelkezésre áll másodlagos memória
Elsősorban egytaszkos működés	Multitaskos működés

1.1. táblázat. Főbb különbségek a PLC-k és a PC-k között [19]

1.3. PLC-k szoftveres felépítése

Ahogy az IEC definíciójában is szerepel, a PLC-t a benne tárolt program teszi arra alkalmassá, hogy ellássa a feladatát. A tárolt program két fő részből áll: az operációs rendszerből és a felhasználói programból.

Az operációs rendszer feladata a belső – PLC-rendszerből független – funkciók biztosítása, pl.: diagnosztika, memória kezelés, kommunikációs feladatok ellátása, felhasználói program futtatása [8]. Az operációs rendszer a PLC védett tárhelyében, a felhasználói programtól elkülönítve van tárolva, valamint a felhasználó által a legtöbb esetben nem módosítható.

A felhasználói program a PLC-rendszer egyik legfontosabb eleme, ez tartalmazza azokat az utasításokat, amik az irányított folyamat és a hozzá kapcsolódó gépek viselkedését meghatározzák [8]. A felhasználói program és a hozzá kapcsolódó felhasználói adatok tárolására a PLC-ben szintén külön memória terület áll rendelkezésre.

A PLC programozásának egyik nagy előnye, hogy számítógép segítségével végezhető. A PLC gyártók által biztosított alkalmazások, grafikus felületet, fejlett programhiba-keresési lehetőségeket, projekt menedzselési szolgáltatásokat és egyéb program fejlesztő eszközöket tartalmaznak.

1.3.1. PLC programok szerkezete

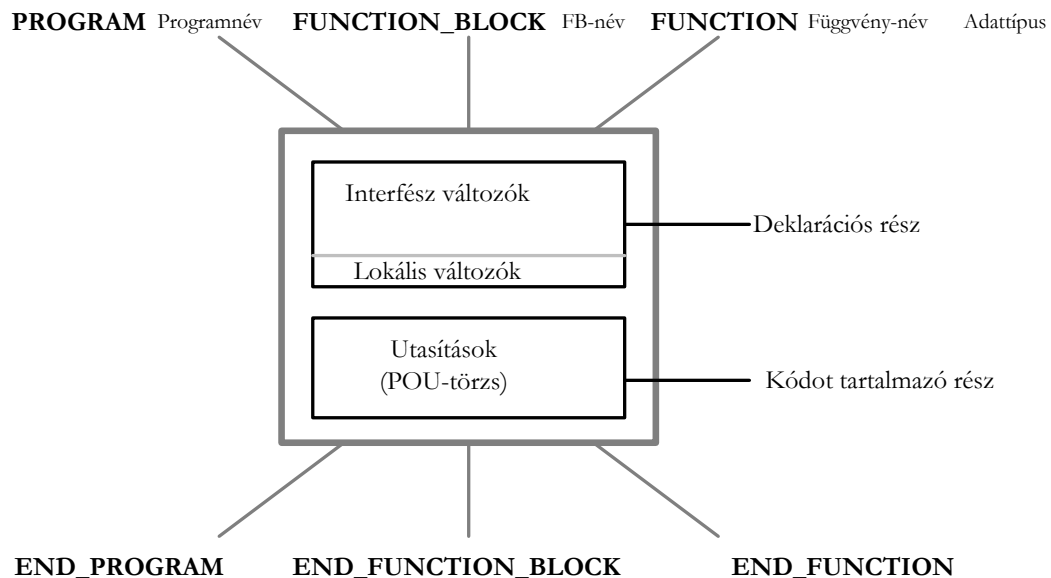
A PLC-k programozására 1993-ig – szabványos nyelvek hiányában – csak az egyes gyártók által kínált gyártóspecifikus nyelvek léteztek. Ez nem csak a szintaktikában, de az egyes nyelvekben írt programok szerkezetében is eltéréseket okozott. Ezt megoldandó a legjelentősebb gyártókat tömörítő PLCopen szervezet 1993-ban jelentette be az IEC 61131-3 szabvány első kiadását. A szabvány a PLC-programok szerkezetére vonatkozóan meghatározza a program

1.3. PLC-K SZOFTVERES FELÉPÍTÉSE

szervezési egység (Programszervezési egység – Program Organization Unit (POU)) fogalmát: „a legkisebb független egysége a felhasználói programnak” [9]. A szabvány ajánlásait követve a programokban előforduló POU-k típusai:

- Függvény: a legegyszerűbb POU, nem rendelkezik hozzárendelt memóriával, emiatt belső állapotát nem tudja eltárolni. Ez azt jelenti, hogy egy adott bemenetre mindig ugyanaz a kimenet a válasza.
- Funkció blokk – Function Block (FB): rendelkezik saját változókkal, melyeket automatikusan hoz létre az interpreter az FB első meghívásakor (instanciálás). Ezekkel a változókkal egy FB képes eltárolni a belső állapotát.
- Program: a felhasználói programot reprezentáló POU [9]. Szélesebb I/O kezelési jogosultságokkal rendelkezik, mint egy FB.
- Szervező blokk – Organization Block (OB): a felhasználói programok meghívását végző POU.
- Adat blokk – Data Block (DB): adatstruktúrák tárolására szolgáló POU.

A POU-k általános szerkezetét mutatja az alábbi ábra:



1.2. ábra. Egy POU felépítése. A deklarációs rész csak szöveges nyelven adható meg. [9]

1.3.1.1. PLC-k programozási nyelvei

Az IEC 61131-3 szabvány a PLC-k programok készítésére három grafikus és két szöveges programozási nyelvet ajánl. Ezek használata egységesebb szerkezetű programokat és a különböző PLC gyártók platformjai közötti nagyobb átjárhatóságot biztosít. A szabványosított nyelvek általában egy-egy speciális alkalmazási területből fejlődtek ki.

Létra Diagram – Ladder Diagram (LD). Grafikus programozási nyelv, ami a relés vezérlések áramútervéből fejlődött ki és ahhoz hasonló jelölési rendszert használ. Egyszerűbb vezérlési feladatok megoldására előnyös az alkalmazása, mivel szemléletes és könnyen áttekinthető. Bonyolultabb feladatok esetén azonban az alkalmazása nehézkes.

Funkcióblokk diagram – Function Block Diagram (FBD). Szintén grafikus programozási nyelv, de szerkezetében a logikai hálózatokra hasonlít. A programot elemi blokkok és I/O változók hálózata építi fel. A blokkok alkalmazása lehetővé teszi előre definiált elemek (pl.: számlálók, időzítők) használatát.

Sorrend vezérlési diagram – Sequential Block Chart (SFC). A sorrendi vezérlési diagram a harmadik grafikus programozási nyelv az IEC szabványban, ami a Grafcet programozási nyelvből fejlődött ki és a szerkezete számítógépes flow-charthoz hasonlít. Célja a hardver-orientált programnyelvek (LD, FBD) korlátozott programszervezési lehetőségeit kiegészíteni. Az SFC használata tehát a struktúrált programfejlesztést segíti, ezért bonyolultabb feladatok esetén előnyös az alkalmazása [1].

Struktúrált szöveg – Structured Text (ST). Szöveges programozási nyelv, ami magas szintű utasításokat és programszervezési lehetőségeket tartalmaz. Szerkezetében a Pascal és a C nyelvekre hasonlít. Segítségével bonyolult számításokat végző programok készíthetők.

Utasításlista – Utasításlista – Instruction List (IL). Az utasításlista alacsony szintű szöveges programnyelv, amely a Assembly nyelvre hasonlít. Az ezen a nyelven írt program utasítások sorozatából áll és soronként egy utasítást tartalmaz. Szerkezete emiatt egyszerű, segítségével a hardvert jól kihasználó program írható [9].

2. fejezet

Ipari automatizálási rendszerek

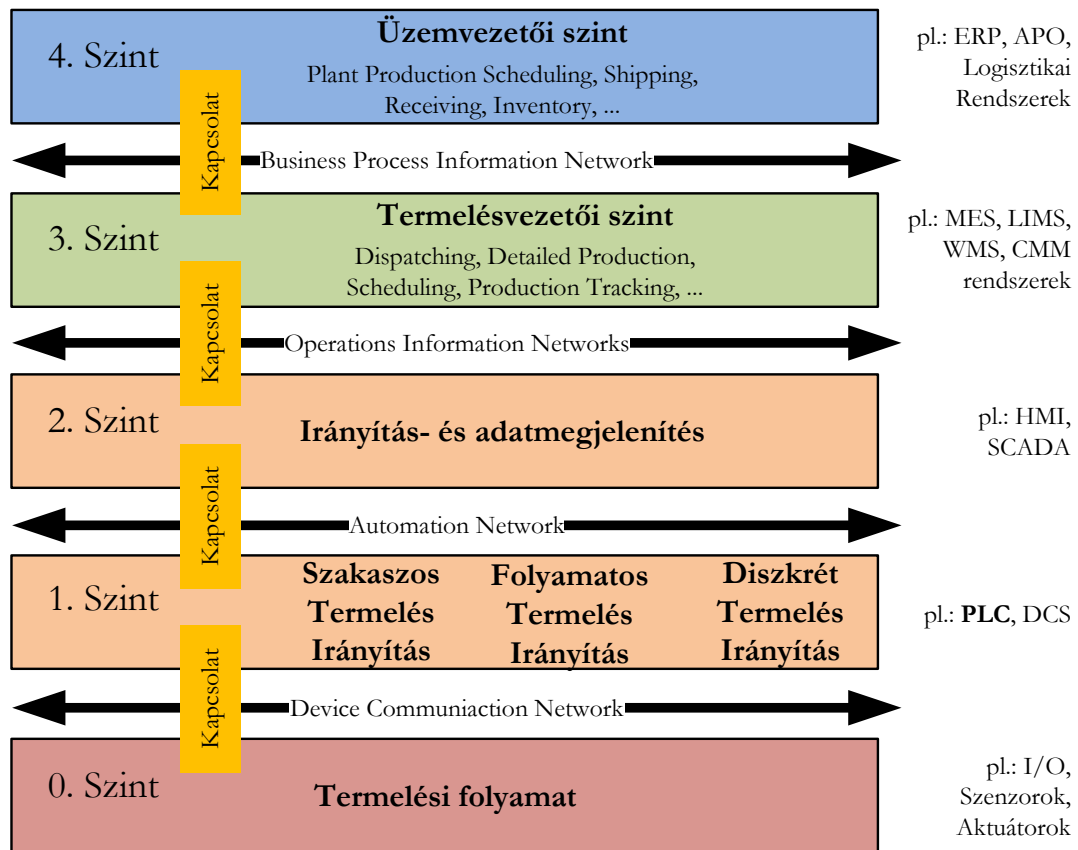
Ez a fejezet ismerteti az ipari automatizálási rendszerek felépítését, valamint a PLC-k szerepét az ipari automatizálási rendszerekben.

2.1. Az ipari automatizálási rendszerek elemei

Az irányítási feladatok egy bizonyos bonyolultsági foka után az irányítási feladatok ellátására egy PLC már nem mindig elegendő [1]. Ezen kívül biztonsági, megbízhatósági és feladatszervezési megfontolások miatt is gyakori, hogy az irányítási feladatokat összekapcsolt PLC-k hálózatával valósítják meg [2]. Az így létrejövő automatizálási rendszer működéséhez kapcsolódó feladatok megoldásához további, magasabb szintű vezérlések lehetnek szükségesek (pl.: az összekapcsolt PLC vezérlése egy fővezérlővel). Az ilyen többszintű automatizálási rendszer szerkezetének szemléltetésének gyakori eszköze az ún. automatizálási piramis [22].

Az automatizálási piramis használatával az ipari automatizálási rendszer egymással kapcsolatban lévő szintekre bontható fel. Ezzel a szemlélettel az egyes szintek és az azokon elhelyezkedő eszközök szerepe könnyebben definiálható, valamint a szintek közötti kommunikáció jobban szemléltethető.

Az automatizálási rendszerek szintjeit vonatkozó definíciókat az ISA/ANSI 95 és az IEC 62264 szabványok tartalmazzák. A 2.1 ábra az ezek alapján elkülönített szinteket és a köztük lévő kapcsolatokat szemlélteti.



2.1. ábra. Az ipari automatizálási rendszerek szintjei. Megjegyzés: ábrán látható 1. és 2. szint alkotja az automatizálási szintet. [4] alapján.

2.2. PLC-k szerepe az ipari automatizálási rendszerekben

A 2.1 ábrán látható, hogy a PLC az automatizálási rendszer első szintjén helyezkedik el, tehát közvetlen kapcsolatban a saját szintjén (automatizálási szint) és az információs (harmadik) illetve a nulladik (eszköz) szinteken lévő eszközökkel van.

Eszköz szint. Az ezen a szintén található eszközök közvetlenül a gyártási folyamathoz kapcsolódnak. Működésükre a nagy számú Input / Output (I/O) művelet jellemző, ezért ezen a szinten nagy megbízhatóságú és valós idejű működésű kommunikáció szükséges. Ezt a leggyakrabban ún. terepi buszok alkalmazásával biztosítják.

Automatizálási szint. Az automatizálási szinten a PLC-k, CNC-k és egyéb vezérlő berendezések közötti kommunikáció zajlik. Ezen a szinten is elvárás a nagy megbízhatóságú, valós idejű kommunikáció, de az eszköz szinthez képest már egy magasabb absztrakciós szintű kommunikáció zajlik (nagyobb adatcsomagok, ritkább kommunikáció [22]). A kommunikációt a terepi buszok magasabb szintekhez kifejlesztett változatai biztosítják

A PLC-k szintje fölött helyezkedik el, de még az automatizálási szinthez sorolhatóak a különböző Supervisory Control and Data Acquisition (SCADA) és felhasználói interfész (Human Machine Interface (HMI)) rendszerek. A SCADA rendszerek olyan számítógépes alkalmazások, melyekkel a vezérléshez kapcsolódó adatgyűjtési és beállítási feladatok végezhetőek el. Az összekapcsolt úgynevezett HMI-SCADA rendszerek használata szolgáltatások széles választékát biztosítja: diagnosztikai eszközök, adat- és folyamat-vizualizáció, diagram megjelenítés, naplózás, alkalmazásfejlesztés [6].

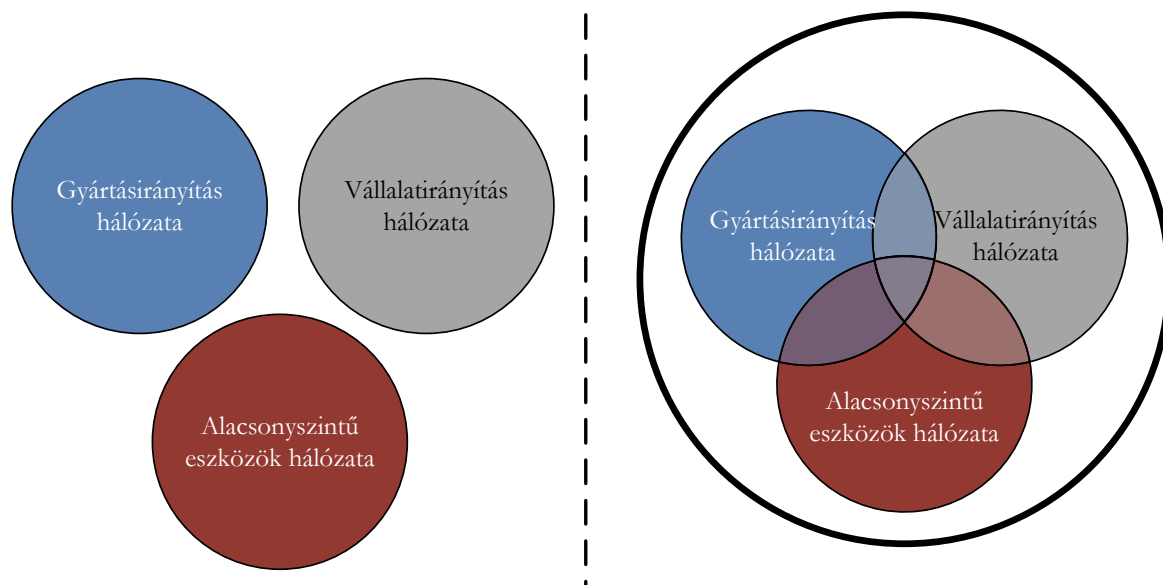
Információs szint. Az információs szinten az gyártásszervezéssel kapcsolatos stratégiai döntésekhez szükséges eszközök találhatóak. Ezen a szinten már nem elvárás a terepi buszok által nyújtott szolgáltatások, ezért a leggyakrabban Ethernet alapú hálózat található.

2.3. Az ipari automatizálási rendszerek felépítésének fejlődési iránya

A információs technológiák fejlődésével az ipari automatizálási rendszerek alsóbb szintjein lévő eszközök is egyre fejlettebb szolgáltatásokat nyújtanak és intelligensebb működésre képesek. Ez azt teszi lehetővé, hogy az eddigi struktúra helyett már az alsóbb szinteken is megjelenik a magasabb szintekre jellemző működés (pl.: adatfeldolgozás) [22].

Ezzel párhuzamosan a hálózati kommunikációs eszközök és szabványok fejlődése is zajlik, amelynek egyik iránya egy egységesebb hálózati struktúra bevezetése. Ehhez egy olyan hálózati protokollra van szükség, ami egyszerre képes kiszolgálni az összes szinten felmerülő igényeket. Ilyen hálózati protokoll az ipari Ethernet, melynek alkalmazása lehetőség nyit egy új, az irányítási szinteket összekötő integrált hálózati struktúra létrehozására.

Ennek használata több szempontból is előnyös az eddigiekhez képest: a konzisztens hálózati struktúra költséghatékonyabb, flexibilisebb valamint az adat-integritás miatt biztonságosabb.



2.2. ábra. Balra a hagyományos hálózati struktúra (elkülönülő alkalmazási szintekkel), jobbra az egységes hálózati struktúra látható [5].

3. fejezet

Szimuláció

Ebben a fejezetben a modern szimulációs eljárások és alkalmazásuk előnyeinek rövid ismertetése után az automatikus szimulációs-modellépítés kerül bemutatásra.

3.1. Bevezetés

Szimuláció fogalma alatt egy olyan technikát értünk, amivel valós vagy elképzelt rendszerek viselkedése vizsgálható [15]. A vizsgálat elvégzésére a legtöbb esetben a rendszer modelljén kerül sor. Modellnek nevezzük a rendszer egy olyan egyszerűsített vagy idealizált (gyakran matematikai eszközök általi) leírását, amelynek célja számítások és előrejelzések készítésének megkönnyítése [The Oxford English Dictionary]. A rendszer tulajdonságait egy adott pillanatban leíró változókat a rendszer állapotának nevezik [12]. Szimuláció során a rendszer matematikai modelljén végzünk kísérleteket [15].

A szimuláció során megkülönböztetünk statikus és dinamikus szimulációs modelleket, aszerint, hogy a rendszer egy adott pillanatbeli reprezentációja vagy a viselkedés időbeli lefolyása a vizsgálat célja [12]. A rendszer állapotváltozóinak időbeli változása alapján beszélhetünk diszkrét vagy folytonos rendszerekről illetve azokhoz tartozó modellekről [12].

3.2. Gyártórendszerek szimulációja

A gyártórendszerek általánosságban dinamikus diszkrét idejű rendszerekként modellezhetők, bár ez nem jelenti azt, hogy kizárólag ilyen tulajdonságú folyamatok és elemek alkotják [10]. A gyártórendszerek szimulációjához ezért a leggyakrabban alkalmazott módszer a DES [12].

A gyártórendszerek működése számos aspektusból vizsgálható szimuláció segítségével, pl.: gyártásütemezés, kapacitás-elemzés [16] valamint lehetőség van a szimulációs eredményeken

alapuló statisztikai elemzések készítésére [17].

A gyártórendszerek szimulációs vizsgálatának előnyei a következők[17]:

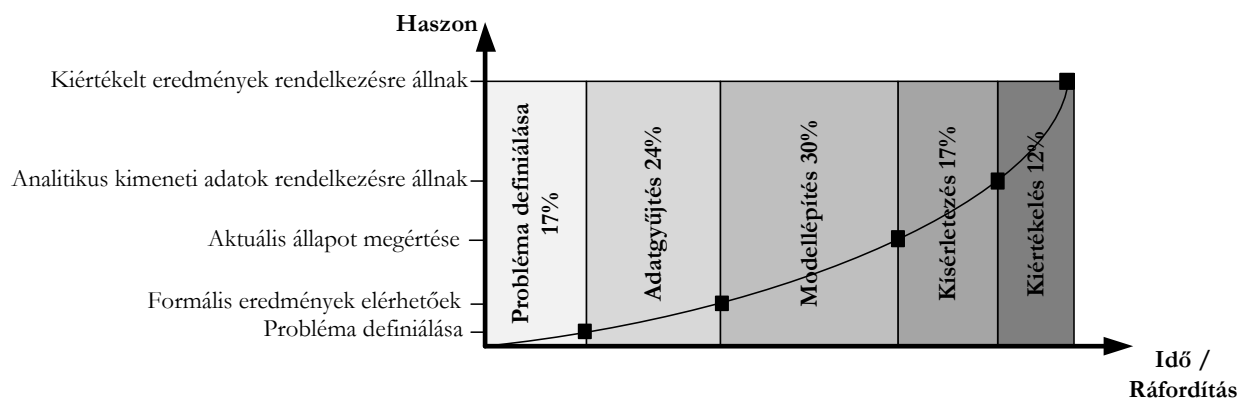
- A vizsgálat eredményét felhasználva a teljesítmény növekszik, csökkennek az átfutási idők, a raktárközi készletek és a működési költségek.
- A modellezés elősegíti a rendszer jobb áttekinthetőségét.
- Lehetőség nyílik „virtuális” statisztikák készítésére.

Ezekon felül a szimuláció segítségével a magasabb automatizálási szinteket segítő, fejlett döntéstámogató rendszerek létrehozása lehetséges [17].

3.3. Automatikus szimuláció-építés

Habár a DES használata széles körben elterjedt a gyártórendszerek szimulációs vizsgálatára, a használhatóságát csökkenti a tradicionális szimulációs és modellépítési lehetőségek korlátai. A szimulációs-modell építése általában lassú és erőforrásigényes feladat. Ennek egyik legfőbb oka a szükséges adatgyűjtés nehézségei [16]. További hátrányos tulajdonság, hogy a szimulációs modell létrehozása egy adott pillanatbeli állapotnak megfelelően történik, valamint a bemenetét a legtöbbször historikus adatok képezik, ezért az idő múlásával a szimuláció relevanciája csökken [21]. Ennek elkerülésére a szimulációs modell gyakori frissítése szükséges, viszont ez szintén idő- és erőforrásigényes feladat [21].

A szimulációs vizsgálat egyes folyamatainak idő- és erőforrásigényét szemlélteti a 3.1 ábra. Látható, hogy a modellépítés és a kapcsolódó adatgyűjtés erőforrásigénye jelentős.



3.1. ábra. A szimulációs folyamat idő- és erőforrásigénye [17].

A fentiekben felsorolt okok miatt igen nagy előrelépést jelent a szimulációs-modellépítés folyamatának automatizálása. Ehhez egy olyan adatforrásra van szükség, amiből az automatikus modellépítéshez szükséges adatok kinyerhetők. Ilyen forrásként szolgálnak az automatizált gyártórendszerek alacsony szintű vezérlőiben tárolt adatok. Az alacsony szintű adatokból épített szimulációs modell egy új modellezési aspektus, mivel a jelenleg elterjedt modellezési módszerek magasabb absztrakciós szinten működnek [16], [7]. Egy ilyen rendszer alkalmazása az automatikus szimuláció-építés előnyein kívül egy a tradicionális szimulációs modellekben nem szereplő szinttel is kiegészíti a szimulációs modellt, szélesítve ezzel a szimulációs vizsgálatok spektrumát [16].

4. fejezet

A megvalósított feladat ismertetése

A dolgozat témája egy jelenleg folyamatban lévő kutatási projekthez kapcsolódik, melynek célja egy gyártóüzem szállítórendszer-modelljének DES-rendszerben végzett automatikus modellépítése (rekonstrukciója). A megvalósítás első lépésében a szállítórendszer fizikai kapcsolati struktúrájának (topológiájának) rekonstrukcióját kellett elvégezni. Az alábbiakban egy olyan módszert fogok ismertetni, melynek segítségével az alacsony szintű vezérlésekben tárolt adatok és programok alapján a topológia automatikusan visszafejthető.

A módszer előnye, hogy a nagyméretű szállítórendszer modelljének felépítése és – változtatások esetén – a frissítése a hagyományos modellépítési technológiákkal hosszadalmas és erőforrás igényes feladat.

4.1. A tárgyrendszer bemutatása

A modell-rekonstrukció megkezdéséhez szükséges a szóban forgó rendszer szerkezetének alapos ismerete. Ez ebben az esetben egy görgőspályás szállítórendszer, melynek hossza közel 5 km. A nagy kiterjedés már önmagában is indokolja, hogy a gyártórendszerrel kapcsolatos szimulációs vizsgálatok valamilyen formában tartalmazzák a szállítórendszer modelljét, hiszen egy ekkora rendszer igen nagy mértékben befolyásolja az egész gyártórendszer teljesítményét.

Egy ilyen nagy méretű szállítórendszer vezérléséhez több PLC összekapcsolt hálózatára van szükség. A görgőspálya-rendszert több kisebb egységre, körzetre bontva vezérlik. Minden körzet vezérlési struktúrájának tetején egy PLC helyezkedik el, ami az adott körzetben található pályaelemeket vezérli, valamint kapcsolatot biztosít a szomszédos körzetek és az összes körzet fölött elhelyezkedő fővezérlő PLC felé. A körzeteket vezérlő PLC-k a szomszédos körzetekhez és az alsóbb szinteken lévő irányítási eszközökhöz terepi buszhálózaton

keresztül kapcsolódnak, a fővezérlő PLC-hez pedig Industrial Ethernet hálózaton keresztül.

A szükséges vezérlési programok és adatok tehát a körzeteket vezérlő PLC-kben kerülnek tárolásra. Ezek hozzáférhetőek PLC operációs rendszer és a programfejlesztő-szoftver szolgáltatásait használva. Az IEC 61131-3 szabvány által ismertetett projekt struktúrájának megfelelően, a pályarendszer vezérlő programjai is POU-kból épülnek fel. A program az alábbi POU típusokat tartalmazza, hierarchiai sorrendben:

- Szervező blokk – Organization Block (OB)
- Program
- Funkció blokk – Function Block (FB)
- Funkció csoport – Function Group (FG)
- Függvény - Function
- Adat blokk – Data Block (DB).

A szükséges adatokhoz való hozzáférés érdekében fontos a vezérlő program szerkezetének és a kapcsolódó pályarendszer ismerete, mivel így állapítható meg, hogy mely POU-k exportálására van szükség. Ezek tartalmazzák azokat az adatokat, amelyek alapján a szimulációs modellt felépítő objektumokat lehet majd létrehozni és tulajdonságaikat beállítani. A pályarendszert az alábbi elemtípusok építik fel:

- Görgőspálya Elem (GE): a szállítórendszerben a leggyakoribb továbbítóelem, görgők segítségével továbbítja a rakományt.
- Szíj Elem (SZE): a GE-khez képest keresztirányú mozgást biztosítja.
- Emelőasztal Elem (EE): a GE és a SZE között továbbítja a rakományt.
- Lift Elem (LE): a görgőspálya szintjei között szállítja a rakományt.
- Forgóasztal Elem (FE): a rakomány orientációját képes megváltoztatni.

Az exportáláshoz tehát azokat a POU-kat kell kiválasztani, amelyek a fent említett elemtípusok működését határozzák meg, illetve a POU-k feldolgozásához szükséges információt hordoznak. Ezeket az adatokat minden körzet vezérlő PLC-jéből exportálni kell. Ami így az alábbiakat jelenti:

- FB-k, amelyek az egyes elemekhez tartozó FG-kat tartalmazzák.

- DB-ok, amelyek az elemek paramétereit tárolják.
- Szimbólum tábla, ami a programban használt változók memória címeit tartalmazza.

4.2. Kódelemzés

A kiválasztott adatok exportálásánál figyelembe kell venni, hogy a további feldolgozás melyik exportálási formátumban végezhető el a leghatékonyabban. Mivel a POU-k mindegyike az IEC 61131-3 szabvány által definiált nyelvek valamelyikében íródott, ezért ezek átfordíthatók egymásba, így akár szöveges, akár grafikus formában exportálhatóak a szükséges POU-k. Fontos megjegyezni, hogy a változó deklarációkat és adattáblákat tartalmazó programrészeket, a választott programozási nyelvtől függetlenül, szöveges formában lehet csak leírni. Emiatt, valamint a grafikus elemeket tartalmazó nyelvek feldolgozási nehézségei (pl. képfeldolgozási eljárások alkalmazása) miatt, exportálási formátumnak a Utasításlista – Instruction List (IL) nyelvet választottam.

Az így kapott kódfájlok tartalma megegyezik a PLC fejlesztő környezetben látható tartalommal, azonban nem tartalmaz olyan további leíró információt (pl. XML-tageket), amit a kódfeldolgozás során a struktúra felismerésére lehet használni. Emiatt a kódfeldolgozás következő lépése, hogy az exportált kódot egy szövegelemző (ún. *Parser*) alkalmazással kell analizálni. Egy ilyen alkalmazás a ProGrammar.

4.2.1. Kódelemzés a ProGrammar környezet segítségével

A ProGrammar egy olyan vizuális környezetet biztosít, ami alkalmas platform független Parserek létrehozására valamint biztosítja a hozzájuk tartozó Grammar Definition Language (GDL)-t. A GDL egy magas-szintű jelölési rendszer, amit adatok szerkezetének leírására lehet használni [13]. Egy GDL-ben definiált grammatikát (*Grammar*-t) használva, a Parser alkalmazás képes arra, hogy egy adott típusú adatsort elemezzen.

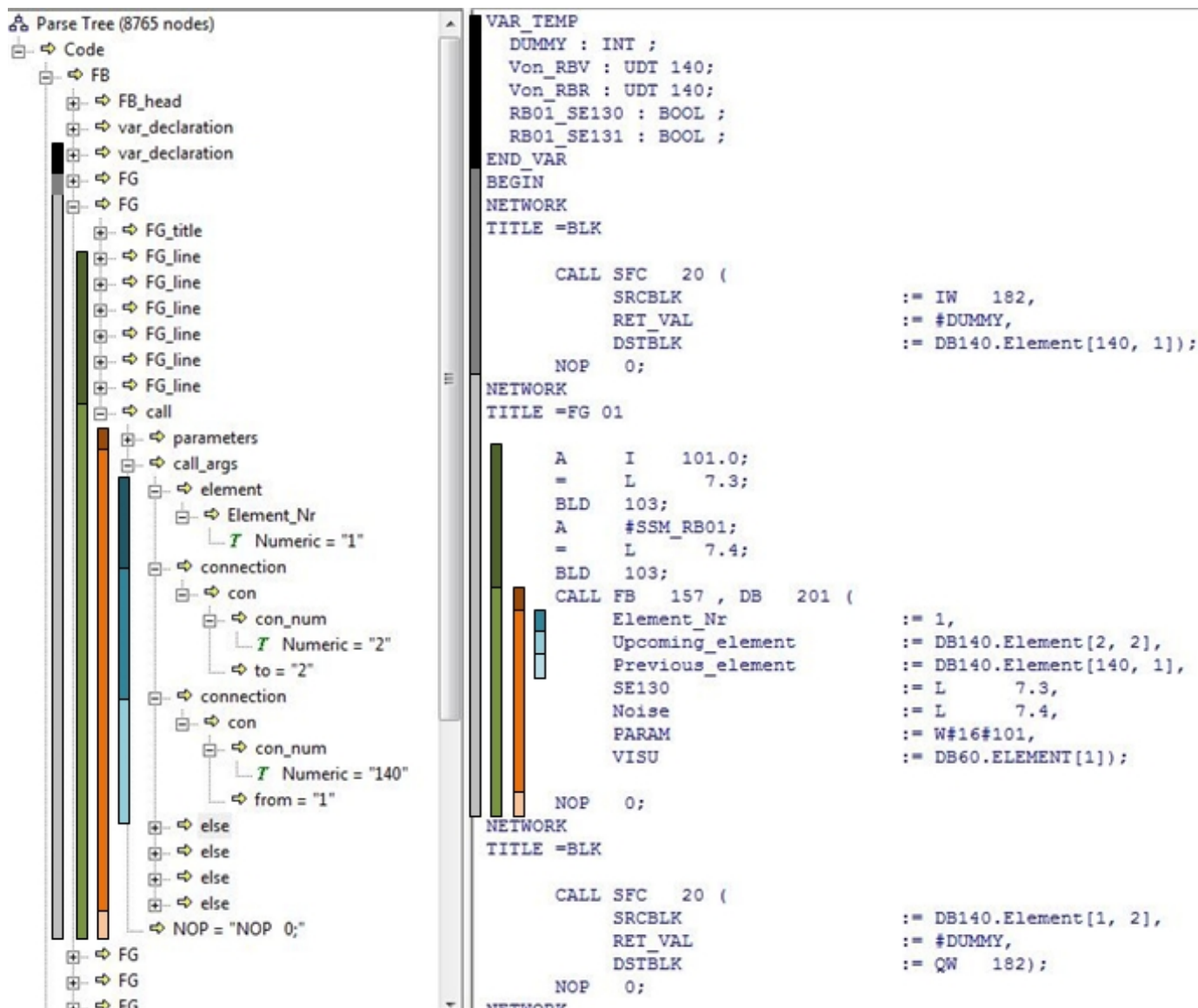
Egy grammatikának tartalmaznia kell minden olyan szabályt és szimbólumdefiníciót, ami ahhoz szükséges, hogy a nyelvtani elemzés végrehajtható legyen. Az elemzés és a további feldolgozás annál hatékonyabban végezhető el, minél inkább tükrözi az adott grammatika az elemzendő kód struktúráját. Az exportált plc-kód fájlokhoz tartozó grammatikák készítésekor így több szempontot is figyelembe kellett vennem:

- A grammatika legyen univerzális, képes legyen kezelni az egyes kód fájlok eltéréseit.
- Jelenjen meg a topológia-építés szempontjából lényeges információ.

- A későbbiekben felmerülő további elemzési szempontok alapján, könnyen módosítható, kiegészíthető legyen. (Pl.: logikai kapcsolatok felépítéséhez szükséges információk)

4.2.2. A kódelemzés eredménye

A kódelemzés eredménye egy fastruktúra, amiben megjelennek a nyelvtanban definiált elemek a – szintén a nyelvtanban definiált, a PLC-kód felépítését tükröző – szerkezetnek megfelelően. Egy ilyen elemzés eredményét szemlélteti a 4.1 ábra.



4.1. ábra. Az ábrán látható, hogy a kód (jobbra) topológia szempontjából szignifikáns részei (türkiz árnyalatok) nagyobb terjedelemben jelennek meg a fában (balra).

A 4.1 ábrán látható kódrészlet az alábbi részeket tartalmazza:

- Változó-deklarációs rész (*fekete*).
- Adatblokk másolást végző rendszerfüggvény (*sötétszürke*).
- Pályaelemhez tartozó FG (*világosszürke*).

A változó deklarációs rész tartalmazza az adott FB által felhasznált globális és lokális változókat.

Az adatblokk másolást végző rendszerfüggvény egy olyan előre definiált függvény, aminek segítségével az egyes memóriaterületek között lehet adatmásolást végezni. A topológiában található elágazások megvalósításában fontos szerepe van, ezért már a grammatika készítés és kódelemzés során is figyelembe kellett venni, hogy megjelenjen a fastruktúrában.

A pályaelemhez tartozó FG két részre osztható: logikai utasításokat tartalmazó részre (*sötét zöld*) és egy FB hívásra (*világos zöld*). Az FB hívás a CALL paranccsal történik és egy előre definiált FB instanciálását eredményezi. A meghívandó FB azonosítója és a hozzá tartozó adatstruktúra megnevezése a CALL parancs után található (*barna*). Ezután az FB instanciáláshoz szükséges bemenő paraméterek átadása következik (*narancssárga*), amit a hívást lezáró null utasítás követ (*világosbarna*).

A FB bemenő paraméterei között türkizzel jelölt részek olyan információkat tartalmaznak, amiket felhasználva az rekonstruálandó rendszer topológiája felépíthető. Ez az alábbiakat jelenti:

- Az FB-hez tartozó elem sorszáma, ami egy egyedi azonosítóként használható a szállítórendszer körzetén belül (*sötét türkiz*).
- Az adott elemhez kapcsolódó következő szállítóelemet megnevező változó (*türkiz*). Ez a változó szintén tartalmazza a megnevezett elem egyedi azonosítóját, valamint még a kapcsolat irányultságára vonatkozóan is hordoz információt. A következő elem felé mutató kapcsolatot „2” jelöli, míg az előző elem felől jövő kapcsolatot „1”.
- Az adott elemhez kapcsolódó előző szállítóelemet megnevező változó (*világos türkiz*). A változó típusa megegyezik az előbb ismertetettel, így a 4.1 ábrán látható világos-türkizzel jelölt kapcsolat jelentése: fizikai kapcsolat a 140-es elem felől.

A fent bemutatott hivatkozási rendszerben a görgőspálya minden egyes eleme rendelkezik egy egyedi azonosítóval, ami egy 1, 2 vagy 3 számjegyből álló szám. A modellépítés során azonban az egyedi azonosításon kívül lényeges információ a pályaelemek típusa is (4.1-ben

ismertetett típusok). Ehhez olyan adatstruktúrát kell létrehozni, amiben az egyes azonosítók mellé a hozzá tartozó elem típusa is fel van tüntetve. Ennek alapjául olyan DB-k szolgálnak, amik a PLC programban létrehozott elemekhez tartozó adatokat tárolják. Ilyen DB-k elemzéséhez külön grammatika szükséges, amely képes felismerni a 4.1 fejezetben ismertetett elemtípusokat.

4.3. Az elemzett kód feldolgozása

A ProGrammar környezet nemcsak a grammatika fejlesztésére nyújt lehetőséget, hanem az elemzés elvégzésére is. A elemzés eredménye azonban az előbbiekben ismertetett „címkézett” fastruktúra. Ahhoz hogy a modellépítéshez szükséges adatokat struktúrált formában lehessen kinyerni azonban további lépésekre van szükség, egy olyan alkalmazás segítségével, ami az alábbi feladatokat képes megoldani:

- Képes hozzáférni az kódelemzés eredményéhez
- Megfelelően képes kezelni a kódelemzés eredményeként kapott struktúrát
- Képes előállítani a kívánt kimeneti formátumot (a szimulációs modell számára)

A fenti feladatokat elvégzésére alkalmas a Mathematica nevű matematikai programcsomag. A ProGrammar program biztosítja, hogy az elemző modulja elérhető legyen külső alkalmazás számára, egy alkalmazásprogramozási felületen (Application Programming Interface (API)) keresztül. A Mathematica programot a ProGrammar add-on-nal kiegészítve lehetőség nyílik a ProGrammar szolgáltatásait meghívni, majd további feldolgozást végezni. Ez egyszerűsíti a modell-rekonstrukció folyamatát, mivel nincs szükség két szoftver közötti adatcserére: a kódelemzés a Mathematica programon belül megy végbe.

4.3.1. Kódfeldolgozás a Mathematica programmal

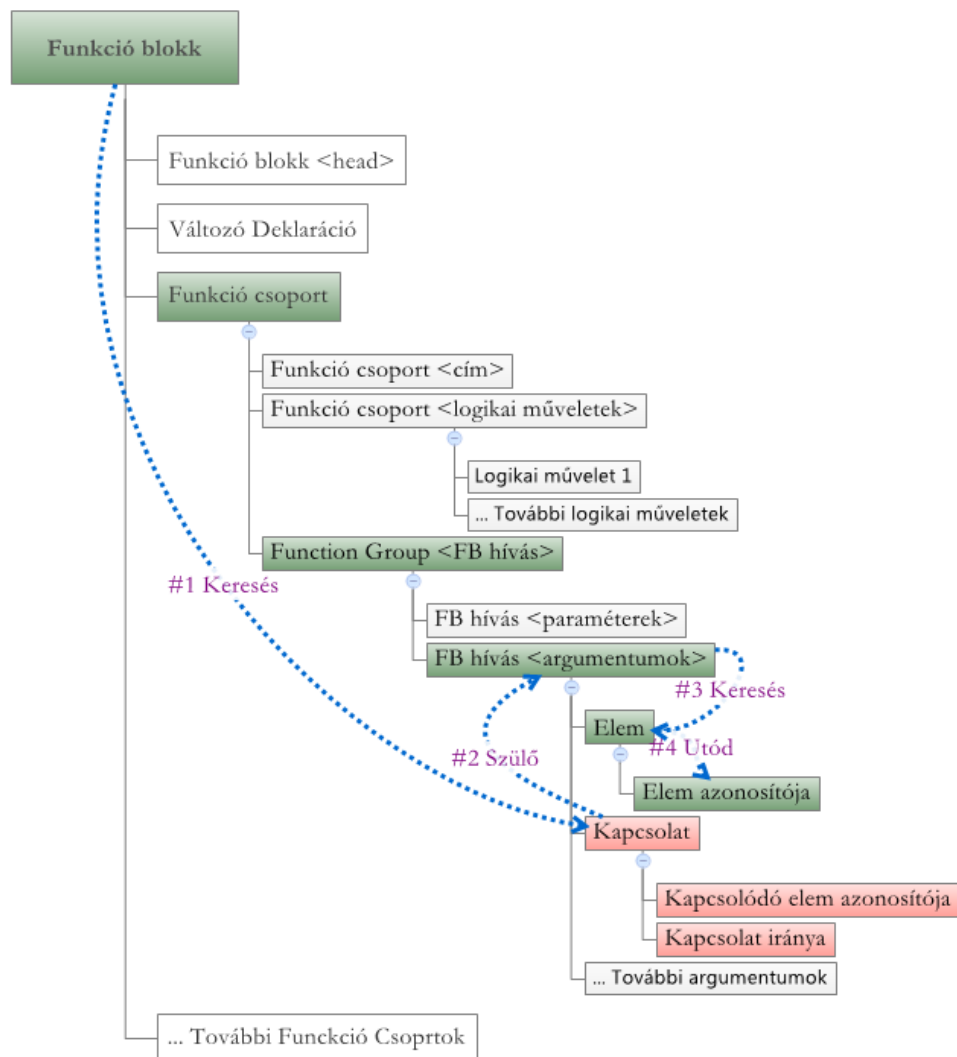
A Mathematica programban a ProGrammar add-on betöltése után be kell olvasni a PLC-kből exportált forrásfájlokat és a hozzá tartozó grammatikát. Ezek után a kódelemzés elvégezhető, eredménye a már ismertetett fastruktúra. A topológia-építés következő lépésében ennek a fastruktúrának az elemeihez kell hozzáférni. Ehhez az alábbi gráftopológiai eszközök szükségesek:

- Utód keresés: a fa egy csúcspontjához tartozó utódok keresése. Egy elem utódai „*az elem kimenő éleinek végződő csúcshalmaza*” [23].

4.3. AZ ELEMZETT KÓD FELDOLGOZÁSA

- Szülő keresés: a fa egy csúcspontjához tartozó szülő keresése. Egy elem szülője „*bejövő élei kezdőpontjainak halmaza*” [23].
- Keresés: adott tulajdonsággal (értékkel, címkével) rendelkező csúcsok keresése. (A keresés hatékonysága növelhető, ha csak egy adott ágban keres az algoritmus)

A 4.2 ábrán látható, hogy a fenti lépések segítségével, hogyan található meg egy pálya-elem azonosítója és a hozzá tartozó fizikai kapcsolatot leíró hivatkozás. A keresés során a kapcsolatot jelző hivatkozásokat kell megkeresni, majd a hozzájuk tartozó elemazonosítókat.



4.2. ábra. A kapcsolatok összegyűjtéséhez szükséges lépések. A nyilaknak és a rajtuk lévő számoknak megfelelő sorrendben.

A fenti ábrán látható a lépéseket a *BuildConnections* nevű függvénybe építettem bele. A függvény kimenete a kapcsolatokat leíró lista, melynek elemei a következő információkat

tartalmazzák:

{ELEM AZONOSÍTÓJA, KAPCSOLÓDÓ ELEM AZONOSÍTÓJA, KAPCSOLAT IRÁNYA}

Példa: {10,12,2}

Az eredményül kapott lista a modellépítéshez még nem elegendő, ugyanis az alábbi hiányosságai vannak:

- Csak az azonosítókat tartalmazza. Az elemek típusáról nem tartalmaz információt.
- Redundáns, mivel egy kapcsolat kétszer is megjelenhet benne: az elemek fordított sorrendben és ellenkező kapcsolati iránnyal.
- A pályarendszer elemeinek hivatkozásai nem minden esetben egyértelműek: a nem görögspálya elem (GE) vagy szíj elem (SZE) típusú elemek azonosítója a kódban 0.
- Elágazást jelölő hivatkozásokat tartalmaz.

A továbbiakban a fenti problémákra adott megoldás kerül ismertetésre.

4.3.1.1. Elemtípus keresés

A 4.2.2 fejezetben már részben bemutatásra került a probléma megoldása. Olyan DB-kat kell az erre a célra megírt grammatikával elemezni, amikből meg lehet állapítani az elemek azonosítóit és típusjelzőit is. Az ilyen DB-kat tartalmazó forrásfájlokon végzett elemzés eredménye egy újabb fastruktúra. Ez azonban lényegesen egyszerűbb struktúra, mivel a grammatika erre a célra van optimalizálva és az elemzés szempontjából lényegtelen részeket nem illeszti be a fába. (Ellentétben a kapcsolatok keresésére használt grammatikával, ahol éppen az volt a cél, hogy minél inkább bővíthető legyen és jobban tükrözze a kódszerkezetet a továbbfejleszthetőség érdekében.)

Az elemzés eredményét a már ismertetett gráftopológiai eszközökkel feldolgozva minden körzethez létrehozható egy olyan lista, ami tartalmazza az adott körzetben előforduló elemek nevét és típusát:

{ELEM TÍPUSA, ELEM AZONOSÍTÓJA}

Példa: {GE,28}

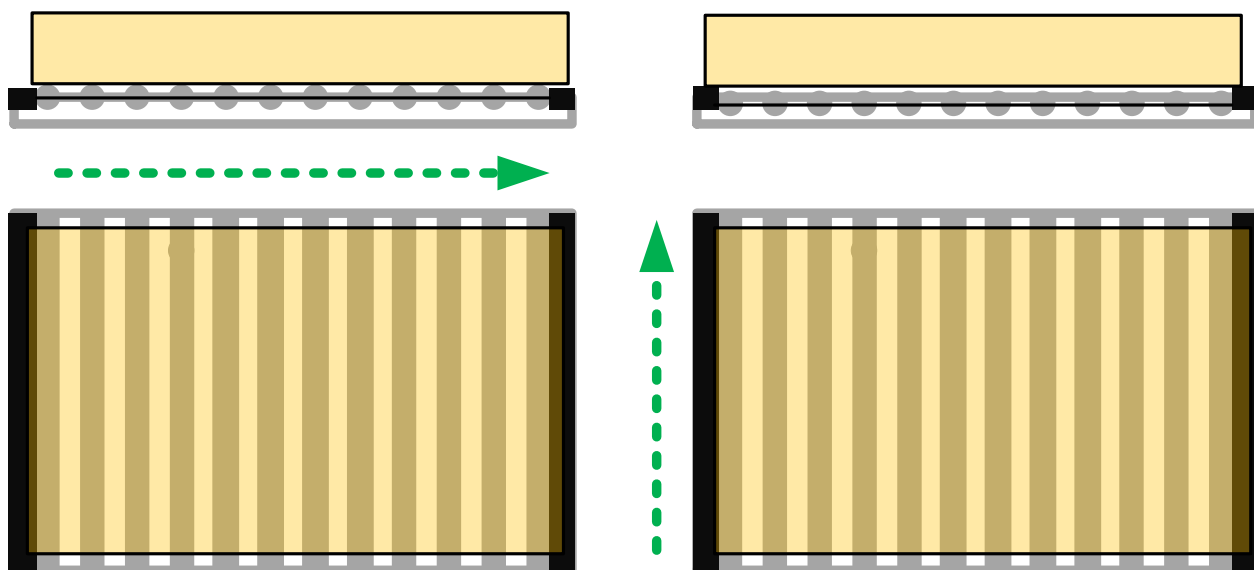
Ezeket a listákat egy *search* nevű függvénnyel kereshetővé tettem. A függvény bemenete a körzet azonosítója és az elem azonosítója, kimenete pedig az elem típusa és azonosítója.

4.3.1.2. Redundancia kezelése

A kapcsolati listában lévő redundancia a modellépítés során problémát okozhat. A megoldás érdekében a tárolt kapcsolatok irányát előre definiálni kell. Ezután a lista minden elemét ennek az iránynak megfelelően kell rendezni, majd eltávolítani az ismétlődéseket. Ezeket a műveletet a *CleanUpConnections* nevű függvénybe építtem bele. A függvény bemenete egy kapcsolatok tartalmazó lista, kimenete pedig egy redundanciát nem tartalmazó lista.

4.3.1.3. Azonosítatlan elemek kezelése

A kódban szereplő azonosítatlan elemek esetében a pályarendszerről rendelkezésre álló ismereteket kell a kódfeldolgozásban felhasználni. Egy EE ugyanis mindig más elemek egy meghatározott környezetében fordul elő, pl.: GE $\rightarrow$ EE $\rightarrow$ SZE. Ez egy ún. sarokelem, aminek a felépítését a 4.3 ábra mutatja be.



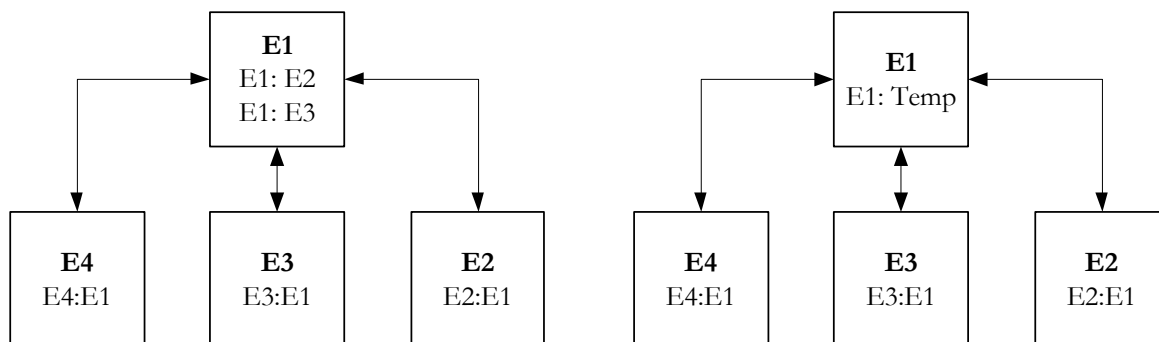
4.3. ábra. Egy sarokelem működési állapotai és a továbbítás irányai: a baloldali ábrákon a továbbítást a Görgőspálya Elem (GE) végzi, a jobboldali ábrákon pedig a Szíj Elem (SZE). Az Emelő Elem (EE) feladata a görgősor emelése és süllyesztése.

Az EE tehát fizikailag csak a két szomszédos elem számára elérhető. Az ilyen sarokelemekhez tartozó elemmintákat előre definiálva, lehetővé válik egy olyan lista készítése, ami tartalmazza az előforduló sarokelemeket. Ezt a *FindCornerElements* függvénybe épített eljárásokkal oldottam meg. Ezután ezt a listát úgy kell kereshetővé tenni, hogy egy adott elem

azonosítatlan kapcsolatait megadva bemenetként, a tábla alapján azonosított hivatkozást adja vissza. Ezt végzi a *searchEE* függvény.

4.3.1.4. Elágazások kezelése

A PLC-kód szerkezeti kötöttségei miatt – ideiglenes változók használata nélkül – minden elemhez maximum két másik elemet lehet közvetlenül kapcsolni. Ez – főként a kapcsolatok irányítottsága miatt – gyakran nem elegendő. (Pl.: Ha egy elem 3 másik elemmel is kétirányú kapcsolattal kapcsolódik, akkor az elemenkénti 2 hivatkozás nem teszi lehetővé a kapcsolat teljes leírását.) Ezt a problémát a PLC programban átmeneti változók használatával oldották meg: a kapcsolati hivatkozást egy átmeneti változó jelöli, ami különböző logikai feltételek teljesülése esetén más és más kapcsolati hivatkozás értékét veszi fel.



4.4. ábra. Az elágazások megvalósítása a PLC-kódban. Látható, hogy a baloldali ábrán *E1* elem egyik kapcsolata nem jelenik meg. A jobb oldali ábrán alkalmazott *Temp* változó a logikai feltételektől függően felveheti *E2*, *E3*, *E4* értéket.

A topológia-építés szempontjából azonban a logikai feltételek irrelevánsak: ha két elem összeköttetésbe kerülhet, akkor ott lennie kell fizikai kapcsolatnak. Ezért elegendő azt megvizsgálni, hogy az adott átmeneti változó milyen értékeket vehet fel. Ezt a kódban található, a vizsgált átmeneti változónak értéket adó, adatblokk másoló rendszerfüggvények elemzésével lehet elvégezni.

A Mathematica programban az átmeneti változókhoz tartozó kapcsolati hivatkozások összegyűjtéséhez szükséges műveleteket a *BuildTempVarList* nevű függvénybe építettem bele. Ez a függvény egy az átmeneti változókat tartalmazó listát ad eredményül, amit a *searchTempVar* függvénnyel tettem kereshetővé, aminek használatával a kapcsolati listában lévő átmeneti változók az eredeti hivatkozott értékekre cserélhetők.

4.4. Adatszerkezet a modellépítéshez

A kódfeldolgozás eredménye – a kiegészítő függvények alkalmazása után – minden elemzett körzethez egy-egy kapcsolati lista, ami a körzetben lévő összes kapcsolatot leírja. A modellépítéshez ezen felül az alábbiakra van szükség:

- A felhasznált elemek listája. Ez fog az objektum építés alapjául szolgálni a DES rendszerben.
- A megjelenítendő elemek koordinátái. Mivel a DES rendszer nem tudja automatikusan elhelyezni a létrehozott objektumokat, ezért a vizualizáció érdekében, a megjelenítendő elemek koordinátáit előre meg kell határozni.

A topológia szemléltetéséhez szükséges koordináták meghatározásához a Mathematica *GraphPlot* gráfmegjelenítőjét használtam. A kapcsolati információkból felépített gráf tükrözi a pályarendszer valós fizikai topológiáját, így az ebből nyert koordináták használata indokolt. A Mathematica-ban felépített gráfot és arra épülő DES-rendszerben lévő modellt a 4.5 ábra mutatja be.

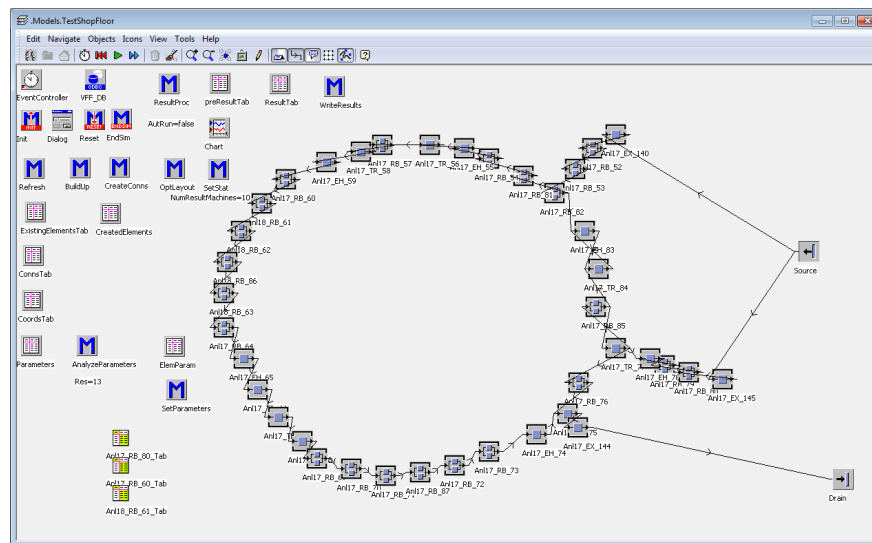
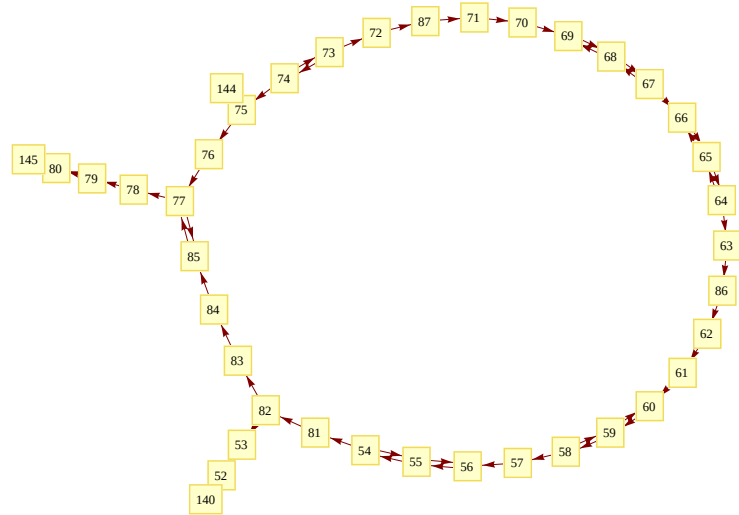
SQL-tábla neve	Tábla oszlopainak neve			
Létrehozandó elemek	Körzet azonosítója	Elem neve és azonosítója	-	-
Példa	K7	GE10	-	-
Kapcsolatok	Körzet azonosítója	Elem neve és azonosítója	Kapcsolódó elem körzet-azonosítója	Kapcsolódó elem neve és azonosítója
Példa	K10	GE12	K10	SZE8
Koordináták	Körzet azonosítója	Elem neve és azonosítója	X-koordináta	Y-koordináta
Példa	K6	EE23	10,25	32,00

4.1. táblázat. Az SQL adatbázisban létrehozott táblák.

A kódfeldolgozás eredménye tehát három lista (táblázat), amiket a DES rendszer számára elérhetővé kell tenni. Erre a célra relációs adatbázist használtam, mivel a Mathematica és a

4.5. AZ LÉTREHOZOTT RENDSZER FELÉPÍTÉSE

Plant Simulation program is képes SQL kapcsolatot létesíteni egy relációs adatbázissal. Az adatbázisban felhasznált adattáblák szerkezetét ismerteti a 4.1 táblázat.



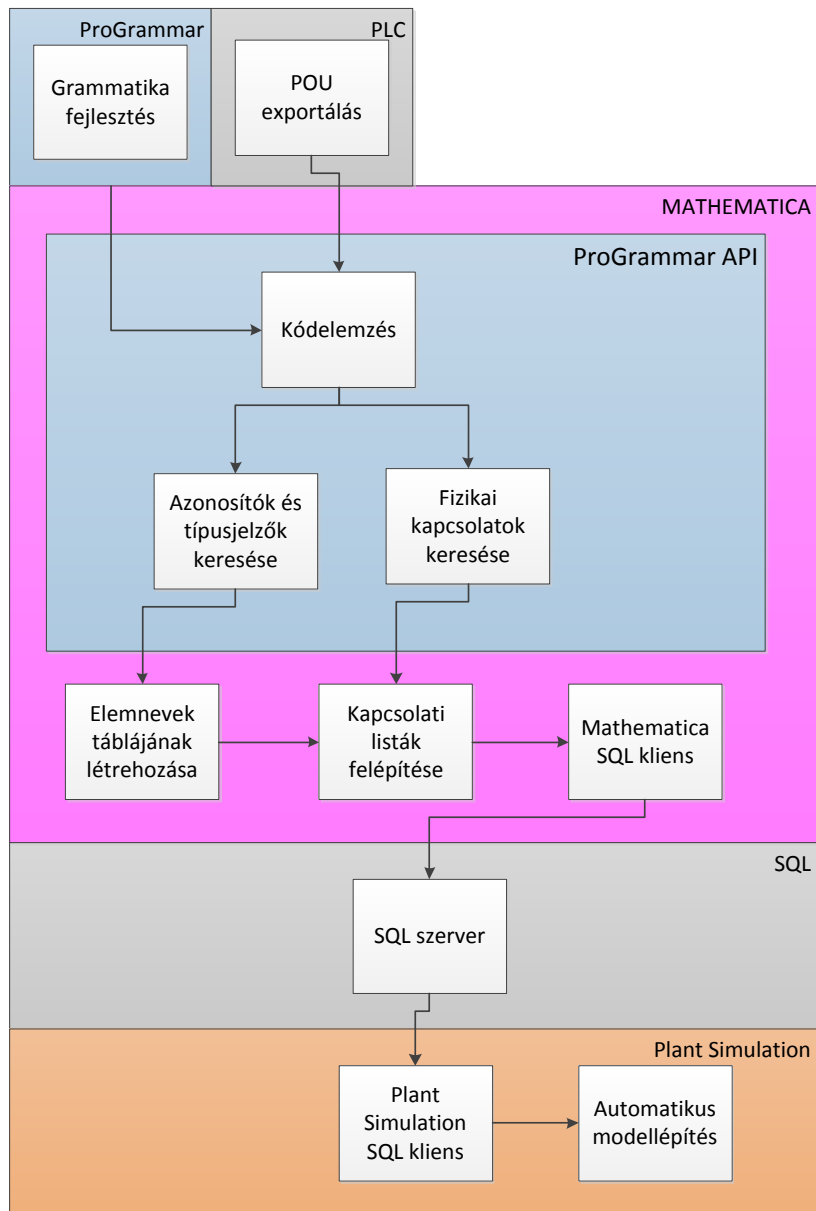
4.5. ábra. A Mathematica-ban előállított gráf (fent) és a gráftopológia elrendezését felhasználó szimulációs modell (lent).

4.5. Az létrehozott rendszer felépítése

A fentiekben bemutatott eljárások együttesen alkotják azt a rendszert, amivel az automatikus modellépítés elvégezhető. A programok funkcióit, egymásba épülését és az adatáramlást a 4.6 ábra mutatja be. Fontos megjegyezni, hogy a végfelhasználás szempontjából a feltüntetett programok közül csak Mathematica programban elkészített függvények használata

4.5. AZ LÉTREHOZOTT RENDSZER FELÉPÍTÉSE

szükséges, mivel a ProGrammar és PLC csak a fejlesztéshez szükséges, az SQL-szerver pedig elérhető a Mathematica SQL-klens segítségével.



4.6. ábra. Az automatikus gyártórendszer modell-rekonstrukciós rendszer struktúrája.

5. fejezet

Összefoglalás, továbbfejlesztési lehetőségek és értékelés

5.1. Összefoglalás és az eredmények értékelése

Napjaink magas automatizáltsági fokú gyártórendszereit, átfogó és nagy bonyolultságú automatizálási rendszerek szolgálják ki, melyek nagy mértékben befolyásolják az egész gyártórendszer teljesítményét. Ezért a gyártórendszerek működésével kapcsolatos vizsgálatoknál is figyelembe kell venni az automatizálási rendszereket. Ilyen vizsgálatok elvégzésére elterjedt módszer a díkszrét idejű eseményvezérelt szimuláció.

A szimulációs vizsgálat egyik kulcsfontosságú lépése a modellépítés és a hozzá kapcsolódó adatgyűjtés, ami a modellezendő rendszer összetettségétől függően, nagy idő- és erőforrásigényű feladat lehet. Az automatizálási rendszerek alacsony szintjeit is figyelembe vevő szimulációs-modell készítése, ezért a tradicionális szimulációs módszerekkel nem vagy csak nehezen kivitelezhető. Erre a problémára jelent megoldást a szimulációs-modellépítés automatizálása.

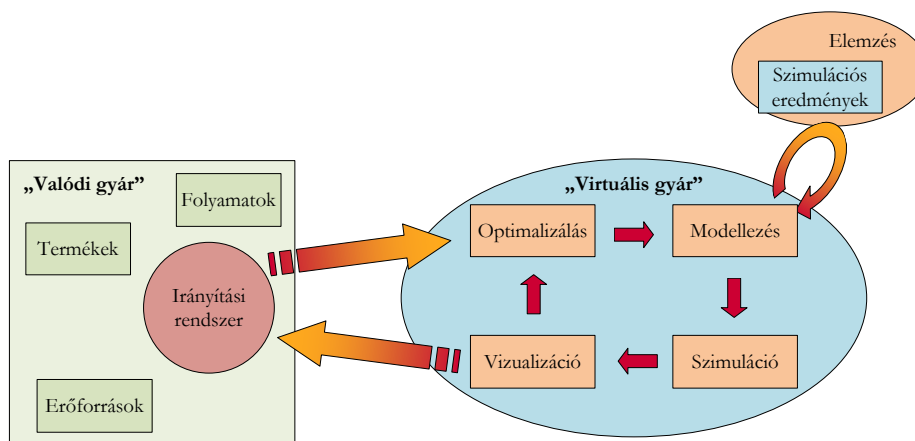
TDK dolgozatomban az automatikus szimulációs-modellépítéshez alkalmazott adatgyűjtés forrása az alacsony szintű vezérlőkben (PLC-k) tárolt adatok és programok. A feladat megoldásához létrehozott rendszer segítségével ezekből az adatokból rekonstruálható egy gyártórendszerben lévő szállítórendszer topológiája. A létrehozott rendszerben kialakításra került egy kapcsolati réteg, melynek segítségével a modell-rekonstrukció eredményét egy DES-rendszer számára elérhetővé lehet tenni és rajta szimulációs vizsgálatokat lehet végezni.

A feladat megvalósítása során sikerült létrehozni egy olyan rendszert, ami képes a vizsgálat tárgyát képező szállítórendszer topológiai modelljének rekonstrukciójára az alacsony szintű vezérlőkben tárolt adatok és programkódok alapján. Továbbá a rendszer felépítését sike-

rült úgy specifikálni, hogy az a továbbiakban bővíthető legyen a teljes modell-rekonstrukció elvégzéséhez.

A DES rendszerben felépülő topológiát a rekonstruált rendszer alaprajzaival összehasonlítva megállapítható, hogy a rekonstruált topológia a valós felépítést tükrözi. Így feltételezhető, hogy az elemek megfelelő felparaméterezése után a szimulációs-modell segítségével eredményes vizsgálatok folytathatók.

Fontos megjegyezni, hogy a modell-rekonstrukciós rendszer az adott szállítórendszerhez erősen kötődik és létrehozása rendkívül időigényes feladat, ami látszólag ellentmond a 3.3 fejezetben tárgyalt indoklásnak. Azonban rendelkezik azzal az előnnyel a hagyományos modellépítéssel szemben, hogy az egyszeri jelentős befektetés után a szimulációs-modell állandó karbantartása és frissítése már lényegesen kisebb ráfordítást igényel. Továbbá a szimulációs modellhez kapcsolódó új alkalmazások lehetőségeit nyitja meg (5.1 ábra).



5.1. ábra. A vezérlési adatok alapján felépített szimulációs modell használata egy „virtuális gyár” környezet elemeként. A valódi és a virtuális gyár közötti folyamatos adatcsere alapja az automatikus szimulációs-modellépítés.

5.2. Továbbfejlesztési lehetőségek

A rendszer továbbfejlesztésének egyik lehetséges területe a szimulációs absztrakciós szintet befolyásoló kódfeldolgozási eljárások alkalmazása. Ilyen eljárással megvalósítható lenne a topológiában intelligens elemösszevonások végzése. Ennek eredményeként a szimulációban szereplő elemek száma csökkenne és a topológia egy meghatározott mértékben egyszerűsödne, így a szimulációs-modell absztrakciós szintje igény szerint változtatható lehetne, amivel a szimulációs futtatás kisebb számítási kapacitást igényel.

Az automatikus modellépítés következő lépése, hogy a topológia rekonstrukciója után a vezérlési logikák visszafejtésére is fel kell készíteni a rendszert. Ez a már meglévő grammatikák további fejlesztését, valamint ezekhez tartozó új feldolgozó algoritmusok létrehozását igényli (esetleg további POU-k exportálását).

További fejlesztési lehetőséget jelent a pályaelemek historikus adatok alapján történő felparaméterezése, melyet PLC-khez kapcsolódó SCADA rendszerekben tárolt adatok segítségével lehet elvégezni. Ehhez azonban egy új adatkapcsolati réteg kidolgozása szükséges, ami a SCADA rendszerek felé biztosít kapcsolatot.

A továbbfejlesztési lehetőségek között megemlítendő még a rendszert alkotó programok további integrációja. A rendszert alkotó programok (ProGrammar, Mathematica, Plant Simulation) mindegyike rendelkezik olyan API-val, melynek segítségével külső alkalmazás számára is elérhetőek a program szolgáltatásai. Az API-k használatával a jelenlegi rendszer felépíthető bármelyik olyan programozási környezetben, ami képes kezelni az API-kat (pl.: JAVA). Ezáltal az említett szoftverekre csak a fejlesztés során lenne szükség, ami lényegesen egyszerűsítene a modellépítést. Továbbá a felsorolt szoftverek mindegyike kereskedelmi termék, ezért a rendszer jelenlegi formában való üzemeltetése, jelentős anyagi ráfordítást igényel. A szoftverek API-jai azonban külön is megvásárolhatóak (jellemzően a teljes szoftver árának töredékéért), így a rendszer ezirányú továbbfejlesztése indokolt.

Irodalomjegyzék

- [1] AJTONYI, I. *Automatizálási és kommunikációs rendszerek*. Miskolci Egyetemi Kiadó, 2003.
- [2] ALI, E. Distributed control systems. King Saud University – Process Control Workshop, March 2002.
- [3] BARTH, M., STRUBE, M., FAY, A., WEBER, P., AND GREIFENEDER, J. Object-oriented engineering data exchange as a base for automatic generation of simulation models. *IECON 2010 - 36th Annual Conference on IEEE Industrial Electronics Society* (November 2010), 1335–1340.
- [4] BRANDL, D., AND OWEN, P. A tutorial on the ansi/isa95 enterprise/control system integration. Presentation, September 2003.
- [5] CISCO SYSTEMS. Intelligent networked manufacturing – ethernet to the factory. Osisoft User Conference, April 2006.
- [6] DANEELS, A., AND SALTER, W. What is scada? *International Conference on Accelerator and Large Experimental Physics Control Systems* (1999), 339–343.
- [7] HAN, K. H. Object-oriented modeling, simulation and automatic generation of plc ladder logic. *INTECH: Programmable Logic Controller* (January 2010), 1–17.
- [8] IEC. Iec 61131-3: Programmable controllers – part 1: General information. Tech. rep., IEC.
- [9] K.-H. JOHN, M. T. *IEC 61131-3: Programming Industrial Automation Systems*. Springer, 2001.

- [10] KÁDÁR, B. *Intelligent approaches to manage changes and disturbances in manufacturing systems*. PhD thesis, Budapest University of Technology and Economics, 2002.
- [11] KOSTEK, T. E. Plc fundamentals – common plc terminology. Purdue University – MET 382: Controls and Instrumentation for Automation, August 2009.
- [12] LAW, A. M., AND KELTON, W. D. *Simulation Modeling and Analysis*. McGraw Hill, 2000.
- [13] NORKEN-TECHNOLOGIES. Programmar. Presentation, 2008.
- [14] OF TECHNOLOGY KHARAGPUR, I. I. Industrial automation and control: Module 4: Programmable logic control systems. Webcourse.
- [15] PAGE, B., AND KREUTZER, W. *The Java Simulation Handbook*. Shaker Verlag, 2005.
- [16] PARK, H.-T., KWAK, J.-G., WAND, G.-N., AND PARK, S. C. Plant model generation for plc simulation. *International Journal of Production Research* 48, 5 (March 2010), 1517–1529.
- [17] PFEIFFER, A. *Novel Methods for Decision Support in Production Planning and Control*. PhD thesis, Budapest University of Technology and Economics, 2007.
- [18] POPOVICS, G., PFEIFFER, A., KÁDÁR, B., VÉN, Z., AND MONOSTORI, L. Plc code processing for automatic simulation model generation. *MAS 2011 – The 10th International Conference on Modeling and Applied Simulation* (September 2011), 155–160.
- [19] RIOS-GUITERREZ, F. Overview of programmable logic controllers (plcs). ECE491 - Design Workshop, 2007.
- [20] SIEMENS. Simatic controllers. Tech. rep., Siemens, April 2011.
- [21] TAVAKOLI, S., MOUSAVI, A., AND KOMASHIE, A. A generic framework for real-time discrete event simulation (des) modelling. *Proceeding of the 2008 Winter Simulation Conference* (2008), 1931–1938.
- [22] VOGEL-HEUSER, B., KEGEL, G., BENDER, K., AND WUCHERER, K. Global information architecture for industrial automation. *Automatisierungstechnische Praxis*.
- [23] WOLFRAM RESEARCH INC. Mathworld. Online Encyclopedia, 2011.