Budapesti Muszaki és Gazdaságtudományi Egyetem
Gépgyártástechnológia Tanszék

Intelligens, nyílt gépvezérlések és tudásmegosztási problémáik gyártórendszerekben

Ph.D. értekezés

Nacsa János
MTA Számítástechnikai és Automatizálási Kutatóintézete
CIM Kutatólaboratórium

Budapest, 2001. május

BUDAPESTI MUSZAKI ÉS GAZDASÁGTUDOMÁNYI EGYETEM

GÉPÉSZMÉRNÖKI KAR

Szerzo neve: **Nacsa János**

Értekezés címe: **Intelligens, nyílt gépvezérlések és tudásmegosztási problémáik gyártórendszerekben**

Értekezés benyújtásának helye: **Gépgyártástechnológia Tanszék**

Dátum: **2001. május 31.**

Bírálók: Javaslat:

1. bíráló neve nyilvános vitára igen/nem

2. bíráló neve nyilvános vitára igen/nem

A bíráló bizottság javaslata:

Dátum:

(név, aláírás)
a bíráló bizottság elnöke

Jelen Ph.D. dolgozat bírálatai és a védésrol készített jegyzokönyv elérheto a Gépészmérnöki Kar Dékáni Hivatalában.

Ph.D. ÉRTEKEZÉS TARTALMI ÖSSZEFOGLALÓJA

Szerzo: **Nacsa János**
Cím: **Intelligens, nyílt gépvezérlések és tudásmegosztási problémáik gyártórendszerekben**

Az elmúlt években nagyon sok kutatás folyt a nyílt gépvezérlésekkel valamint a különbözo mesterséges intelligencia módszerek vezérlésekben való alkalmazásaival kapcsolatban. Az értekezésben összefoglalom az elmúlt tíz évben, ebben a témakörben folytatott kutatásaim eredményeit.

Áttekintettem a különbözo vezérlési feladatokban a mesterséges intelligencia módszerek (ismeret alapú, szubszimbolikus, hibrid) alkalmazásait. Részletesen ismertettem az intelligens ágensek vonatkozó alkalmazásait.

Bevezettem a gyártórendszerekben a logikai kommunikáció három szintjét, és tudásbázisú szimulációval illusztráltam, hogy az egyes kommunikációs üzenetek alapján hogyan elemezheto egy gyártórendszer elosztott intelligens viselkedése.

A különféle nyílt gépvezérlési kutatásokat osztályoztam és összehasonlítottam a legfontosabb három gyártó-független vezérlést az általam kidolgozott módszer segítségével. Megállapítottam, hogy sajnos ezek teljesen inkompatibilisek. Javaslatot dolgoztam ki arra, hogy hogyan érdemes a munkát folytatni egy világszerte elfogadható nyílt gépvezérlés létrehozásáért.

Egy PC alapú robot-programozói és -üzemeltetési rendszer fejlesztése kapcsán definiáltam ívhegeszto robotok számára egy STEP alapú robotprogram nyelvet és egy - az MMS robot társszabványán alapuló - virtuális gyártóeszköz modellt.

Vizsgáltam a nyílt rendszerekbe integrált intelligens megoldásokat. Így gyártóeszközöket modelleztem MMS segítségével tudásbázisú környezetben, és így intelligens cellavezérlot tudtam kialakítani. Egy intelligens ember-gép interfészt is létrehoztam egy nyílt szerszámgépvezérlésben.

Részletesen elemeztem az intelligens CNC-k jelenlegi helyzetét, eközben három szinttel jellemeztem a gépvezérlések belso, strukturális adaptivitását.

Bevezettem a vezérlések tudásszerverét (KSC - Knowledge Server for Controllers), amely különbözo gépvezérlések intelligenciájának a növelését támogatja muhelyszinten. A KSC olyan hálózati eroforrás, ami intelligens algoritmusok hatékony futtatását végzi, és így az intelligens muködés lehetoségét szolgáltatja egyéb rendszerek számára. A KSC koncepció széles köru alkalmazhatóságát igazoltam egy villamos energia-elosztó alállomás operátori tanácsadó rendszerének kifejlesztésével.

ABSTRACT OF Ph.D. DISSERTATION

Author: **János Nacsa**
Title: **Intelligent, Open Controllers and Their Knowledge Sharing Problems in Manufacturing Systems**

In the recent years many efforts have been made in the field of open architecture controllers and in applications of artificial intelligence methods within the different controllers. This dissertation summaries the state of the art and also my work done on these topics during the last decade.

I introduced three levels of logical communication in manufacturing systems, and illustrated with a knowledge-based simulation how the behaviour of the distributed intelligence in a manufacturing system can be analysed based on the different messages.

I classified the different open controller projects and compared in details the most important three vendor neutral initiatives using my methodology. I proved that - unfortunately - they are fully incompatible. I made a suggestion on a possible continuation of a world-wide-accepted open controller.

During the development of a PC based robot programming and execution system I have defined a STEP based robot programming language for arc-welding robots and their virtual manufacturing device model based on the MMS robot companion standard.

I investigated intelligent solutions coupled to different open systems. I modelled manufacturing devices with MMS within a knowledge based environment to build intelligent cell controllers. I also added an intelligent human machine interface to an open machine tool controller.

I examined the current problems, limits and results of the intelligent CNCs.

I introduced Knowledge Server for Controllers (KSC) to support different machine tool/robot controllers to grow their intelligence. KSC provides network-based services of efficient intelligent data processing to the controllers in the workshop. As a proof of the wide availability of the conception a KSC was developed in an operator advisory system of an electrical energy substation.

TARTALOMJEGYZÉK

AZ ÉRTEKEZÉSBEN ELOFORDULÓ RÖVIDÍTÉSEK

rövidítés	jelentése	elofordulása
AC	Adaptive Control	
AC	Axis Control	OSACA
AGV	Automated Guided Vehicle	
AO	Application Object	OSACA
API	Application Programming Interface	
ASS	Application Seervice System	OSACA
AWR-VMD	Arc Welding Robot VMD	PROARC
CAN	Controller Area Network	
CAPP	Computer Aided Process Planning	
CNC	Computer Numerical Control	
COM	Communication Object Management	OSACA
CORBA	Common Object Request Broker Architecture	
DAC	Distributed Architecture Controller	
DCOM	Distributed COM	Microsoft
DNC	Direct Numerical Control	
DSP	Digital Signal Processor	
FIP	Fieldbus International Protokoll	
FSM	Finite State Machine	
GSI	G2 System Interface	
HIKE	HPKB Integrated Knowledge Environment	
HIR	Helyi irányító rendszer	Paks
HMC	Human Machine Control	OSACA
HMI	Human Machine Interface	
HMS	Holonic Manufacturing System	
HPKB	High Performance Knowledge Environment	
IDL	Interface Definition Language	Corba
IMS	Intelligent Manufacturing Systems	
JOP	Japan FA Open Systems Promotion Group	
JVM	Java Virtual Machine	Java
KIF	Knowledge Interchange Format	
KQML	Knowledge Query and Manipulation Language	
KSC	Knowledge Server for Controllers	
LAN	Local Area Network	
LLNL	Lawrence Livermore National Laboratory	
NIIIP	National Industrial Information Infrastructure Protocols	
MACRO	Motion And Control Ring Optical	
MAP	Manufacturing Application Protocol	
MAS	Multi Agent System	
MC	Motion Control	OSACA
MIDL	Microsoft IDL	Corba
MLGO	MMS like G2 Object	
MMC	Man Machine Control	
MMC	Motion Managment Control	OSACA
MMS	Manufacturing Message Specification	
MTS	Message Transfer System	OSACA
NCK	NC Kernel	
NIU	Network Interface Unit	PROARC
OGY	Ott - Grebogi - Yorke	káosz
OKBC	Open Knowledge Base Connectivity	

OMAC	Open Modular Architecture Controller	
OPC	Open Process Control	
OSACA	Open System Architecture for Controls within Automation Systems	
OSEC	Open System Environment for Controllers	
OSEL	OSEC Language	
PLC	Programable Logic Controller	
PROARC	CAD-Based **Pro**gramming System for **Arc** Welding Robots	
PSL	Process Specification Language	
RC	Robot Controller	
RMI	Remore Message Invocation	Java
RMS	Reconfugurable Manufacturing System	
RTU	Real-time Unit	Paks
SC	Spindle Control	OSACA
SCADA	Supervisory Control and Data Acquisition	
SERCOS	**Se**riell **R**ealtime **Co**mmunication **S**ystem	
SPC	Statistical Process Control	
SSQA	Simulation, Scheduling and Quality Assurance	
STEP	**St**andard for **E**xchange of **P**roducts Model Data	
TIME	Total Integrated Manufacturing Enterprise	OMAC
TEAM	Technologies Enabling Agile Manufacturing	OMAC
UDP	User Datagram Protocol	
ÜRIK	Üzemirányító Rendszer Informatikai Központ	
VMD	Virtual Manufacturin Device	MMS

1 BEVEZETÉS

Az informatika évente újabb fogalmakat és élethelyzeteket teremt az élet minden területén. Az információfeldolgozás egyik hagyományos és a kezdetekig visszameno alkalmazási területe a gyártásautomatizálás. A legkülönbözobb új informatikai módszerek, szoftver és hardver termékek kipróbálásakor szinte mindig találni a gyártás egyszerusítését, ésszerusítését vagy hatékonyabbá tételét megcélzó eredményeket és tapasztalatokat. A másik oldalról a gyártásautomatizálás, a maga fejlodésével, folyamatosan generálja a kihívást jelento problémákat.

Sajnos az információtechnológia (IT) és a valódi gyártás problémái között létezik egyfajta szakadék [88], ami miatt nehéz, illetve idonként csak nagy késéssel lehet az IT eredményeit a gyártásautomatizálásban hasznosítani. Bár az IT szinte naponta szállít újabb eredményeket, amelyek jó és járható megoldásoknak tunnek a gyártás megannyi problémájára, az alkalmazások során azonban rendszeresen kiderül, hogy a valós világgal való interfészelés állandó gondot és soha meg nem szuno váratlan problémákat jelent. *Az értekezésem témája ezért a nyílt, intelligens gépvezérlések*.

A **nyílt rendszerek** (4. fejezet) kérdése, igénye az elmúlt 10-15 évben az informatikában rohamosan került elotérbe, többféle értelmezésen esett át, de végül is meghatározó követelménnyé vált a mai elosztott, Internet/Web alapú "e" rendszerek világában. Természetesen a "nyíltság" gyártásautomatizálás sok területén kecsegtet komoly elonyökkel a tervezéstol a gyártásig a gyárinformatika minden szintjén. Túl vagyunk ugyanakkor bizonyos illúziókon is, amelyek szerint a nyílt rendszerek gyorsan elterjednek, és fontos problémákat egyszeruen megoldanak. Ezt illusztrálja a MAP (Manufacturing Automation Protocol) története, amely nem váltotta be a hozzáfuzött reményeket és nem terjedt el a gyártórendszerekben, mint szabványos nyílt kommunikációs felület.

Az értekezésem a nyílt rendszerek segítségével a különféle gyártóeszközök gépvezérlésében született megoldásokat - benne saját eredményeimet - mutatja be, de szót ejt a nyílt rendszerekben rejlo ellentmondásokról, megoldatlan kérdésekrol is.

A **mesterséges intelligencia** (MI) hagyományosan az informatikai érdeklodés, és különösen az alkalmazott kutatás egyik fontos területe, amely újszeru, más módon nehezen megközelítheto, de valós kérdésekre ígér jó válaszokat. Az MI egy gyujtofogalom, sokféle megoldás és módszer sorolható az MI-hez tartozónak (2. fejezet). Az elso MI kísérletek, prototípusok óta rengeteg kutatás foglalkozott az MI alkalmazásával a gyártás legkülönfélébb területein. Kifejezetten jelentosnek minosíthetok a magyar eredmények ezekben a kutatásokban, mintha a gyártásautomatizálással foglalkozó magyar kutatók mindegyikét kisebb-nagyobb mértékben inspirálta volna ez a kihívás.

Egyértelmu és széles körben elfogadott, hogy sok gyártási problémára sikerült MI eszközökkel jó és a gyakorlatban is bevált megoldásokat találni. Számos olyan berendezés, szoftver és nagyobb rendszer található a különbözo gyárakban, amelyeknek egy-egy részletében - többnyire rejtetten - MI algoritmusok futnak. Ugyanakkor a kutatások mennyiségéhez képest - nemzetközi és hazai szinten egyaránt - az ipari alkalmazás elmarad a várttól. Be kell vallani, hogy az elmúlt 30 évben volt legalább 3-4

nagyobb hullám, amikor az MI egyes módszerei olyan jellegu publicitást kaptak, ami irreális várakozásokat keltett. Igaz ez a szimbolikus programozásra, a szakérto rendszerekre, a neurális hálózatokra, szoft-programozásra és az intelligens ágensekre. Célom volt a gyártórendszerekben használt elosztott intelligencia muködésének megértése, a különbözo intelligens rendszerek összekapcsolásából eredo nagyobb rendszer kooperációjának a vizsgálata.

Értekezésemben az MI módszerek alkalmazási területe a különféle (cella, robot, CNC) **gépvezérlések**, amelyek szerepe kulcsfontosságú a gyártástechnológiában. A gyártási folyamat kihagyhatatlan láncszemét jelentik, mert a tényleges gyártást végeztetik el a különféle berendezésekkel. Ennek ellenére egyfajta lemaradás tapasztalható a nyílt és/vagy intelligens megoldások gyakorlati alkalmazásában a vezérlések világában összehasonlítva a gyártástechnológia más - tervezés, ütemezés, minoségbiztosítás stb. - területeivel.

A kutatásom alapveto célja a gép és cellavezérlésekben az MI módszerek alkalmazásával a meglévo rendszerek korlátait meghaladó megoldások keresése, ennek a kérdésnek a módszertani és környezeti feltételeinek a vizsgálata. Ez olyan újszeru megközelítéseket jelent a gépvezérlések körében, amelyekkel a korábbiaknál hatékonyabban lehet integrálni mesterséges intelligencia alapú módszereket a vezérlésekbe. Ehhez alapveto kulcsként a gépvezérlések területén és a muhelyszinten meglévo különféle nyílt rendszeru megoldásokat használtam fel.

1.1 A KUTATÁS AKTUALITÁSA

A kutatás aktualitását az IMS (Intelligent Manufacturing System) programban szereplo projektek vizsgálatával igazolom.

A gyártásautomatizálás területén jelenleg is futó nemzetközi kutatások legjelentosebbike az ún. IMS kutatási keretprogram [84] 1989-ben indult Japánban Yoshikawa professzor kezdeményezésére és 1992-ben vált világméretuvé (USA, Kanada, Európai Unió, stb.). Ebben a programban önmagukban is hatalmas projektek futnak (több millió dolláros költségvetéssel és 10-20 résztvevo vállalattal, kutatóhellyel), és próbálnak korszeru válaszokat adni a gyártás és a hozzá kapcsolódó feladatok (szállítás, szervezés, vállalatirányítás, marketing, környezetvédelem, stb.) olyan akut problémáira, amelyeket világszerte tapasztalnak és küszködnek. Hosszú ido után az elmúlt években sikerült végre magyar kutatóknak is bekapcsolódni az IMS projektekbe.

Tekintettel a program finanszírozási nehézségeire (a kutatási pénzforrásokat résztvevo régiókként kell eloteremteni, a nagy távolságok miatt magasak a járulékos költségek stb.), méltán állítható, hogy csak a legkiválóbb és széleskörü érdeklodésre számot tartó projektek kerülnek be az IMS programba. Az elmúlt évek tapasztalata azt mutatja, hogy általában valamely régió egy-egy sikeres saját kutatása válik világméretuvé. A program nevében szereplo intelligens jelzo nem azt jelzi, hogy kizárólag MI alkalmazásokra kívánnak koncentrálni, ugyanakkor az elnevezés egyértelmuen deklarálja, hogy a jelenleginél sokkal intelligensebb gyárakban és technológiákban gondolkodnak, ahol sokrétuen használják az MI eszközkészletét.

Az IMS egyik elso fontos eredménye volt az 1994-ben készült Megvalósíthatósági Tanulmány [83], mely részletesen ismerteti a jövobeli IMS projektek fobb célkituzéseit, finanszírozását, a kontinenseket átfogó munka szervezését, keretet adva a további munkának. Az itt megfogalmazott legfontosabb célok (lehetové tenni szofisztikusabb muveleteket a gyártás különféle területein; effektívebbé tenni a különféle eroforrások felhasználását; stb.) mindegyike olyan, amelyek - valamilyen vonatkozásban - igénylik a mesterséges intelligencia alkalmazását.

Az 1. táblázatban a futó vagy szervezés alatt álló IMS projektek többségét felsorolom, látható, hogy mindegyikben található az értekezés témájához valamilyen módon kapcsolódó tématerület. A kapcsolódási területeket (1) szerszámgépvezérlés (CNC), (2) mesterséges intelligencia (MI) és (3) kommunikáció (COM) külön jelzem.

a projekt rövidítése	legfontosabb célkituzések	kapcsolódások
IF7 [82]	nagy rendszerek (házak, hajók) automatikus szerelésének intelligens támogatása	MI, COM (autonóm ágensek)
NGMS [127]	jövobeli gyártórendszerek vizsgálata	CNC, MI, COM
Holonic Manufacturing Systems [77],[171]	elosztottan vezérelt és önálló elemekbol épülo gyártórendszerek	CNC, MI, COM
GNOSIS [68]	tervezési és gyártási ismeretek rendszerezése	MI (kvalitatív következtetés)
SIMON [158]	újfajta szenzorok integrálása szerszámgép vezérlésekbe	CNC, MI, COM
HIPARMS [76]	nagyon termelékeny és gyorsan rekonfigurálható szerszámgépek	CNC
STEP-NC [162]	STEP alapú CNC interfész definiálása	CNC
MISSION [115]	globálisan elosztott és virtuális vállalatok tervezésének és muködésének modellezése	COM, MI
INTERGLOBAL [86]	Intelligens interfész modellezésre, szimulációra és optimalizálásra termelés tervezéshez és gyártás vezérléshez	MI
SIMAC [157]	intelligens gyártórendszerek és cellák elosztott tervezorendszere	MI, CNC

1. táblázat Az értekezés témaköreit érinto IMS kutatások

1.2 A KUTATÓMUNKA ÉS AZ ÉRTEKEZÉS HÁTTERE

A kutatás során figyelembe kellett vennem Magyarország, ill. kutatóhelyem szerény súlyát az ipari automatizálás világméretu trendjeinek az alakulásában. Ezért törekedtem, hogy a nagy nemzetközi trendeket és szabványokat értékeljem, felvessem azok elonyeit és hátrányait, és azokhoz illeszkedo megoldásokat keressek.

Az értekezésben leírtakat, azok egymásra épülését magyarázza, hogy a kutatás hátterét és lehetoségét a SzTAKI CIM Kutató Laboratóriumának projektjei jelentették számomra az elmúlt 10 évben.

- A nyílt ipari rendszerekkel, jelesül az MMS-sel, az elso tapasztalatokat az ún. MAP Training Center szoftvereinek és kísérleti cellavezérlojének fejlesztése (1991-1992) során szereztem.

- A G2 szakérto rendszert elsoként az ún. SSQA (Simulation, Scheduling and Quality Assurance) kísérleti ütemezo és minoségbiztosítási szakérto rendszerben használtam (1993), amelyet a SIMAN szimulátorral kellett összekapcsolnom. Erre a munkámra Kovács György, Kopácsi Sándor és Mezgár István támaszkodott értekezéseikben.

- Az elso intelligens nyílt gépvezérlési feladatom az MMS hálózatba kapcsolt robotcella G2 alapú cellavezérlése (1994) volt, ahol felhasználtam Haidegger Géza és társai eredményeit.

- Nyílt ipari interfészek kutatásának a hátterét a PROARC EU projekt szolgálta (1994-1996), amelynek a SzTAKI-ban projektvezetoje voltam.

- Az intelligens, nyílt CNC vezérléssel kapcsolatos kutatásaim elobb az IDAS-OSACA (1996-1998) EU majd a TIME-ICON (1997) - az OMAC egyik realizálása az US-ARMY finanszírozásában - projektekhez kapcsolódott.

- A "vezérlések tudásszervere" (KSC - Knowledge Server for Controllers) koncepció kidolgozásához az alapot a HOSACA OMFB projekt jelentette (1998-1999), ahol Drozdik Szilveszter CORBA alapú rendszerére tudtam támaszkodni. A KSC kísérleti igazolásához pedig a Paksi Atomeromunek kidolgozott tanácsadó rendszer (1998-2000) teremtett lehetoséget, amelynek projektvezetoje voltam.

A kutatásaim szoros kapcsolata a különféle projektekhez egyszerre jelentett elonyt és hátrányt. Több olyan eredményem született, amely csak azért jöhetett létre, mert a CIM Labor be tudott kapcsolódni jelentos nemzetközi kutatásokba (pl. OSACA) vagy a koncepciókat "éles" környezetben ki tudtam próbálni (pl. Paks). A projekt keretein túlmutató ötleteimet, felismeréseimet azonban csak korlátozott mértékben tudtam néhány esetben kidolgozni, mert azok a projekt céljaival nem minden esetben estek egybe (pl. intelligens PROARC).

A kutatómunkám hátterét természetesen leginkább a személyes kapcsolatok jelentik. Elsoként köszönöm feleségemnek, Katinak, akinek szeretete nélkül ez az értekezés nem születhetett volna meg. Köszönettel tartozom munkatársaimnak, akik valóban társaim voltak, munkájukkal, támogatásukkal és észrevételeikkel segítettek: Kovács Györgynek, Haidegger Gézának, Nagy Gergelynek, Daniela Gavalcovának, Mezgár Istvánnak, Kopácsi Sándornak és Drozdik Szilveszternek. Az értekezés egyes konkrét témáiban köszönöm Tömössy Gyula, Schuszter György, Szilvásy Albert, Stephan Peper, Jens Brühl, Peter Lutz, Donald Adrian, Chuck Isreal, Sziklay György, Szederkényi Gábor és Baricz Gábor segítségét. Végül köszönöm azoknak, akiknek példája kedvet csinált ahhoz, hogy ipari automatizálással foglalkozzam: Monostori Lászlónak, Horváth Mátyásnak és Erdélyi Ferencnek.

1.3 AZ ÉRTEKEZÉS FELÉPÍTÉSE

A második fejezetben röviden áttekintem, hogy a különféle MI módszerekkel hogyan oldottak meg az intelligens vezérlések témakörébe eso problémákat. Az értekezés témaköre miatt nagyobb hangsúlyt kap az intelligens ágensekhez kapcsolható megoldások bemutatása. Ebben a fejezetben található egy rövid leírás az értekezésben többször használt G2 intelligens keretrendszerrol [65] is.

A harmadik fejezet bemutatja azokat a vizsgálatokat, amelyek a gyártórendszerek különféle berendezései közötti kommunikáció üzeneteit osztályozza az üzenetekben megjeleno különbözo szintu intelligencia szempontjából.

A negyedik fejezet témája a nyílt gépvezérlések. Ismertetem a különbözo nyílt megoldásokat, és részletesen bemutatom a legfontosabb három nyílt vezérlési architektúrát (OSACA, OSEC-JOP, OMAC). Ezek összehasonlításával igazolom, hogy milyen komoly inkompatibilitási gondok állnak fel a három nyílt vezérlo között. Bemutatom az OMAC jobb használhatóságához kidolgozott eredményemet.

Az ötödik fejezet nyílt vezérlések interfész problémáival foglalkozik, és egy ívhegeszto robot példáján illusztrálva mutatok megoldást. Mind az általam specifikált nyílt robotprogram nyelvet, mind a robot MMS alapú továbbfejlesztett virtuális hálózati modelljét bemutatom.

A hatodik fejezetben tárgyalom azokat a prototípusokat, amelyekben elsoként alkalmaztam a nyílt kommunikációs és gépvezérlési megoldásokat MI eszközök (konkrétan a G2 tudásbázisú rendszer) integrálására cella, programozói munkaállomás és szerszámgép vezérlésekbe.

A hetedik fejezetben foglalom össze az intelligens CNC-vel kapcsolatos elvárásokat, elemzem a jelenlegi IT által nyújtott lehetoségeket, és javaslatot teszek egy újfajta megoldásra.

A nyolcadik fejezet az intelligens CNC-k számára javasolt vezérlések tudásszervere, KSC (Knowledge Server for Controllers) koncepciómat ismerteti. Részletesen bemutatom a koncepció egy más iparágban (villamos iparban) sikeresen megvalósított alkalmazását, valamint egy KSC alapú kísérleti intelligens CNC konkrét kialakítását.

Az összefoglalás ismerteti az értekezésben leírt eredményeimet tézisek formájában, majd röviden beszámolok a munka lehetséges folytatásáról. Az értekezés végén levo irodalomjegyzékbe - külön csoportosítva a saját publikációimat - azokat a cikkeket, Web-linkeket gyujtöttem össze, amelyekre ténylegesen hivatkozom a szövegben. Egyes részek jobb megértését a függelékben közölt specifikációkkal, példákkal, összefüggésekkel igyekszem segíteni.

2 INTELLIGENS VEZÉRLÉSEK LEHETSÉGES MI ALGORITMUSAI

A mesterséges intelligencia információfeldolgozási (gyujtési, rendszerezési, tárolási, átadási, bovítési stb.) módszereket tárgyal, amelyeknek az emberi és az élovilágban megfigyelheto jelenségek, törvényszeruségek a motivációi. Központi feladat a tudásnak a számítástechnikai eszközökkel való megragadása és használata. MI alkalmazások esetében mindig önállóan értelmezni kell, hogy az adott rendszer mitol és milyen értelemben nevezheto intelligensnek.

Az értekezésben használom az intelligens vezérlés és az intelligens CNC fogalmát. Az irodalomban kétféle értelemben szokás ezekre a fogalmakra hivatkozni: (1) informatikai értelemben az intelligens vezérlés mindössze annyit jelent, hogy valamilyen MI algoritmus fut a vezérlés (legalább) egyik moduljában, (2) funkcionális értelemben pedig, azt, hogy a berendezés intelligens módon viselkedik, ami számomra a következoket jelenti:

- Képes az információt, mint tudást - akár bizonytalant is - kezelni, reprezentálni szimbolikus vagy szubszimbolikus módon.

- Képes a rendelkezésére álló tudás segítségével adott feladatokat megoldani keresés, cselekvéssor generálás vagy más MI módszerek segítségével.

Opcionálisan hozzáteszem az alábbi feltételeket, bár ezekkel az értekezés csak érintolegesen foglalkozik:

- Képes alkalmazkodni a változó körülményekhez (adaptív).

- A futási tapasztalatokat képes integrálni (tanulási képesség).

- A muködésének, akcióinak az okát is meg lehet ismerni (magyarázó képesség).

Az értekezésemben az informatikai (1) értelemben használom az intelligens kifejezést, arra törekedve, hogy az adott muszaki feladatban az eredményem elorelépést jelentsen a funkcionálisan (2) intelligens rendszer felé. Az intelligens CNC esetében ennek a kérdésnek a részletes diszkusszióját írtam le a 7. fejezetben.

Az intelligens vezérlések tárgyalása elott érdemes röviden összefoglalni, hogy a különféle MI módszerek ([61, 146] alapján) közül melyek azok egyáltalán, amelyek egy vezérlés on-line muködésekor alkalmasak lehetnek egy-egy (rész)feladat megoldására. Többféle tudásábrázolási, bizonytalanság kezelési, következtetési és tanulási stb. algoritmust kell számba venni.

Az összetettebb, magasabb szintu problémák megoldásához nem kerülheto meg az ismeretek valamilyen szimbolikus ábrázolása. Lehetséges, hogy a rendszerrol egy - a feladat szempontjából hatékony - modell építheto fel. Ugyanakkor a legfontosabb ismeretreprezentációs megoldások (logikai, szabály, keret, szemantikus háló) egyike sem jelent univerzális megoldást. A logikai túl körülményes, a szemantikus háló pedig a keret alapú egyfajta speciális esetének tekintheto, amely grafikai formában reprezentálja az ismereteket. A legtöbb mai rendszerben a keret (gyakorlatilag megfelel az objektum

orientált szemléletnek) és a szabály alapú reprezentáció egyfajta együttélése figyelheto meg. Amit keretben (objektumban) érdemes leírni, azt olymódon, amit szabályban, azt pedig abban írnak le. Ugyancsak megtalálható az ún. procedurális leírás, (bár ennek lehetoségét a hetvenes években még elvetették [61],) amely a deklaratíven kényelmetlenül leírható ismeretek megfogalmazására szolgál.

A bizonytalanság kezelése feltétlenül megoldandó kérdés egy intelligens vezérlésben, bár nem szükséges egyetlen reprezentáció mellett dönteni. A vezérlés alsóbb szintjein egyértelmuen a fuzzy logika, máshol a fuzzy mellett akár a Bayes hálók vagy a Dempster-Shafer modell, vagy a szimbolikus modellek közül a nemmonoton következteto rendszer (nonmonoton reasoning) jöhet szóba.

A következtetés módját meghatározza az ismeretreprezentáció, így a tipikus szabályalapú következtetés, amely lehet cél vagy adatvezérelt, de nagy irodalma és sok sikeres alkalmazása van az alapértelmezésen alapuló következtetésnek (default reasoning), az esetalapú következtetésnek (case based reasoning) vagy a korlátozás kielégítés problémának (constraint satisfaction).

Gyártási környezetben sokféle tanulási eljárás alkalmazása ismert (lásd. pl. Monostori Lászlóék áttekintését [120]). Neurális hálózat alkalmazása esetén meg sem kerülheto a tanulás; ami ugyan a leggyakrabban alkalmazott hálózatok esetében off-line. Az evolúciós számítások (pl. genetikus algoritmusok) alkalmasak nagyon sokféle feladat esetében a lehetséges nagyszámú megoldás közül egy/több "meglehetosen jó" megtalálására. Az on-line tanulással tehát, mint opcionális lehetoséggel számolni kell az intelligens vezérlések esetében is, bár az értekezésben ezzel a területtel nem foglalkozom.

A lehetoségek számbavételekor nem szabad elfeledkezni az MI módszerek hátrányairól és korlátairól sem:

- Körülményes és nehéz egy konkrét MI rendszer verifikálása és validálása összehasonlítva a hagyományos szoftverrendszerekkel.

- "Tudásuk" egy adott szuk területrol származik, emiatt nehéz, sokszor alig megjósolható a viselkedésük olyan események bekövetkeztekor, amelyekrol pl. nincsenek szabályai; nem volt megfelelo minta a neurális háló tanulásakor stb.

- Nincsenek olyan jó és elterjedt módszerek, amelyekkel általánosítható lenne egy-egy konkrét problémára született megoldás rokon feladatokra.

- Komoly számítástechnikai eroforrással és elég nagy válaszidovel kell számolni a legtöbb módszer alkalmazásánál, vagy az alkalmazás egy részében (pl. a neurális háló lassan és nehézkesen tanul, a betanított háló azonban nagyon gyorsan muködik)

Látható, hogy ezek a korlátok szemben állnak az intelligens funkcionalitásról írottakkal, ettol nagyon nehéz a feladat.

2.1 KÜLÖNFÉLE MESTERSÉGES INTELLIGENCIA MÓDSZEREK ALKALMAZÁSAI VEZÉRLÉSEKBEN

A továbbiakban az ismeretalapú, szubszimbolikus és hibrid megoldásokat mutatom be. A modell és ismeret alapú megoldásokat érdemes együtt kezelni, hiszen a konkrét megoldásokban gyakran együtt fordulnak elo, és a számítástechnikai környezettel kapcsolatos jellemzoik hasonlóak. Ezekben az esetekben a szoftver komponensek kialakítása önmagukban nehézkes, ill. csak egy olyan környezetben lehetséges, amely támogatja a módszer(ek) által igényelt ismeretalapú adatfeldolgozást (pl. következtetés, mintaillesztés stb.)

Az intelligens ágenseket részletesebben mutatom be (2.2 alfejezet), mert az értekezésben bemutatott KSC koncepció (8. fejezet) több hasonlóságot mutat az ágensekkel, a ketto részletes összehasonlításra is kerül a 8.2 alfejezetben. Néhány eredmény a 7. fejezetben kerül bemutatása, ahol az intelligens CNC-vel kapcsolatos kutatások jelenlegi helyzetét tekintem át.

2.1.1 MODELL ÉS ISMERETALAPÚ MÓDSZEREK

A 80-as évek végéig a tudásbázisú rendszerek segítségével megvalósított vezérlési feladatok gyártórendszerekben elsosorban ütemezési feladatok kezelését jelentette (pl. [99, 100]), és kevés hangsúlyt kaptak az egyéb vezérlési feladatok. Ennek oka az, hogy az akkori eszközökkel (sebesség, memória stb.) csak munkaállomáson vagy nagy gépes környezetben futottak olyan tudásbázisú rendszerek, amelyek valódi feladatok kezelésére elégségesek voltak. Valódi feladat itt a felhasználandó szabályok, adatok számát, a megoldandó probléma nagyságát, komplexitását érzékelteti, vagyis amelyeket egy humán szakérto nem látna át egyszeruen. Ugyanakkor az ütemezési feladatokra nagyon sok és sikeres megoldás született tudásbázisú megoldással (pl. [99, 100]).

A 90-es években pedig - a tudományos irodalmat megvizsgálva - az MI folyóiratokban ill. konferenciákon kifejezetten kevés a szakérto rendszerekrol szóló cikk összevetve más MI megoldásokkal (neurális háló, fuzzy rendszerek, genetikus algoritmus stb.). A kutatásoknál ez egyfajta sikertelenséget, másfelé fordulás tendenciáját illusztrálja. Ehhez képest meglepo, hogy 1995-ben a teljes MI piac 275 millió dolláros forgalmából 196 millió dollárt tettek ki az ismeretalapú eszközök [61]. Ezek segítségével pedig - sokféle más éles ipari alkalmazás mellett - gyártási környezetekben sok cella- és muhelyszintu vezérlést alakítottak ki (pl. [85]);

Szóhasználati zavarral is lehet találkozni. Sokszor szakérto rendszernek neveznek olyan funkcionálisan tanácsadó rendszereket, amelyekben semmilyen MI módszer nem található. Így pl. Hung-ék [80] fejlesztettek ki egy rendszert, amely H profilú acél elemek hegesztésekor a kiindulási rudak szeletelését hivatott optimalizálni úgy, hogy minél kisebb legyen a hulladék. Hiheto, hogy komoly megtakarítás volt elérheto a gyakorlatban, de a rendszer mégsem nevezheto szakérto rendszernek informatikai szempontból.

Bayes háló alapú reprezentációt használó szakérto rendszer segítségével lehet szenzorok mérési adatait hitelesíteni [81]. Emberi tévesztések és mulasztások kiküszöbölésére használnak ún. megbízhatóság alapú karbantartásban (Reliability Centered Maintenance) szakérto rendszert, pl. dízel motoros buszok esetében [163].

Sikeresen alkalmaztak tudásbázisú rendszert szerszámgép alkotóelemeinek élettartam becslésére, váratlan események, devianciák hatásának elsimítására, gyártócella szisztematikus diagnosztikájának elvégzésére [168].

Diagnosztikai célokra többfelé használnak eset alapú következteto rendszereket [61], ezeknél az a kritikus, hogy milyen mennyiségu korábban felgyult információ áll a rendelkezésre.

Több alkalmazásban egy keretben többféle szabálybázist építettek össze komplex feladatok kezelésére. Így pl. a SzTAKI SSQA rendszere [5] gyártórendszerek ütemezési, szimulációs és minoségbiztosítási feladataira alkalmas.

A vezérlésekben modellezést használó megoldások esetében a Petri hálók [139] alkalmazása a leggyakoribb. Több modell alapú kutatás célozza meg újrafelhasználható kontroll szoftverek eloállítását (pl. [129]).

A tudásalapú rendszerek alkalmazásának jelenlegi korlátai között vannak olyanok, amelyek kiküszöbölhetoek lennének, és ezzel az alkalmazhatóságuk egyszerubbé és olcsóbbá válna:

- Mind a mai napig nem létezik megfelelo szoftvertechnológia tudásbázisú rendszerek építésére, csak az általánosság szintjén mozgó módszertani ajánlások, amelyek közül a CommonKADS módszertan [46] a legelterjedtebb.

- Az egyes tudásbázisok nagyon kis mértékben újrafelhasználhatók, a szoftvertechnológia egyéb területein bevált komponens alapú építkezés itt még csak csírájában jelent meg.

- Nincsenek akár de-facto szabványok ismeretalapú nyelvekre. Az MI hagyományos nyelvei a Lisp és a Prolog bizonyos értelemben túl alacsony szintuek ma már, ezek után pedig már csak az egyedi szoftver környezetek egyéni nyelvi megoldásai léteznek, amelyek a természetes nyelv felé mozdulnak el (pl. G2) vagy valamely más programnyelv bovítésével keletkeznek (pl. CLIPS).

2.1.2 SZUBSZIMBOLIKUS MÓDSZEREK

Kifejezetten gyakoriak a neurális hálókat és/vagy fuzzy logikát használó megoldások a különbözo intelligens vezérlési kutatásokban.

Többféle **neurális hálózatot** használtak sikeresen vezérlési, identifikációs és felügyeleti célokra [141, 168]. A neurális háló alapú vezérlok elonye, hogy (1) nagyon egyszeruen lehet sok bemeneti jelet (szenzor információt) feldolgozatni vele; (2) a háló gyors válaszideje; (3) jó tanulóhalmaz választással bonyolult leképezéseket is képes megtanulni; és (4) nincs szükség részletes vezérlési algoritmusra. A legtöbb esetben elorecsatolt hálózatot tanítottak back propagation (BP) algoritmussal, vagy a BP valamilyen gyorsított tanulási eljárásával.

Neurális vezérlésekbol többféle ismert [141], így pl. az Albus féle CMAC, Kawato hierarchikus és Psaltis többrétegu neurális vezérlése. A CMAC-ban csak a kimeneti rétegben változnak a súlyok, így egyszeru a tanuló algoritmusa. Ismeretes olyan CMAC

alkalmazás, ahol a háló önállóan szabályoz, és olyan is, ahol egy állandó erosítésu vezérléssel párhuzamosan muködik. Kawato hierarchikus vezérlésében a két elorecsatolt hálózat egyike a rendszer dinamikáját, a másik az inverz kinematikáját tanulja meg. Többféle automatikus tanulási eljárással kísérleteztek (Goldberg, Miller stb.) többkimenetu vezérlések esetén (pl. többcsuklós robotkar).

Kifejezetten sok alkalmazásban tanítanak meg neurális hálókat a megmunkálás során fellépo devianciák meghatározására [168] (pl. Warnecke a forgácsolási körülmények beazonosítására, Dornfeld esztergálási, Monostori László [13] marási folyamatok esetén a szerszám állapotának meghatározására használ elorecsatolt neurális hálózatokat, emellett Teshima a vágószerszám hátralevo élettartamát, Monostori a kopottsági állapotát becsli neurális hálóval).

Nagy marógép homérsékleti deformációjának kompenzálásához használ Revilla olyan hálót, amely a mért homérsékleti jelekbol állít elo tengely független deformációs adatokat. Ezekbol és az aktuális tengely pozíciókból egy többváltozós lineáris regressziós algoritmussal kapja meg a tengelyekhez tartozó kompenzációs értékeket [168].

Adott megmunkálás általános modelljének a felépítésére szolgáló neurális háló bevezetésével [121] megoldható, hogy ugyanaz a háló jelentsen kiindulási alapot különféle technológiai feladatok megoldásához.

Fuzzy alapú szabályozás gyakran segít olyan vezérlési problémákban, amikor a nem MI alapú kontroll megoldások (PID, nemlineáris visszacsatolás, LQG, H_∞ stb.) valamiért nem hatékonyak. Ennek többféle oka lehet: nincs megfelelo modell a vezérelt rendszerrol; a rendelkezésre álló jelek bizonytalanok, hiányosak; gyorsabb vagy kisebb számítási kapacitást igénylo szabályozást kell választani stb. Mindazonáltal a fuzzy szabályozás általános tapasztalata azt mutatja, hogy legjobb egy hagyományos (tipikusan PID) szabályozás finomabb beállítására használni fuzzy logikát [156].

Robotvezérlok esetében a fuzzy logika használata az alábbi területeken lehetséges [156]:

- közvetlen szabályozás a szervo körben (sok alkalmazás ismert, pl. Mamdani, Hirota, Watanabe),
- elofeldolgozás, vagyis a magasabb szintu szabályozás számára a szenzor jelek fuzzy feldolgozókon keresztül jutnak el,
- felügyelet és finombeállítás,
- trajektória és mozgás tervezés, az inverz kinematika megoldásával kijött megoldások közötti választás,
- akadály kikerülés, alkatrész helyének meghatározása.

Négy kereku mozgó robot fuzzy vezérlésével Schuszter György foglalkozott [150]. Fuzzy alapú szabályrendszerrel választották ki a forgácsolási körülményeket, felügyelték a munkadarab homérsékletének a változását vagy a munkadarab felületi minoségét [168].

Nagy kultúrája van az ún. **neuro-fuzzy rendszereknek**, amikor megkísérlik a két módszer elonyeit ötvözni. A fuzzy modell kialakításánál a tagsági függvények megadása sokszor nem könnyu, hiszen ezeknek nincs olyan elméleti háttere, mint pl. a

valószínuségnek. Ilyenkor segít, ha neurális hálók tanulásával alakítják ki a fuzzy tagsági függvényeket, szabályhalmazt.

Westkämperék neurális háló segítségével a köszörülési folyamat paramétereinek és a munkadarab minoségi jellemzoinek összefüggéseit vizsgálták. A háló kimeneti eredményeit a korábban meglévo szakértoi tudással összevetve határozták meg a rendszerhez fejlesztett fuzzy szabályrendszer tagsági függvényeit [168].

Kategorizálás kérdése, hogy a **káosz** elméletet MI módszernek tekintik-e. Mivel a legtöbb gyártási folyamat a bonyolultsága miatt analitikai modellel csak bizonyos korlátokkal modellezheto, így a káosz elmélet segítségével is megkísérelheto a gyártási problémák megközelítése. A gyártás különbözo szintjeire (szerszámgép és munkadarab dinamikája; nagy sebességu marás; forgácsképzodés, chatter stb.) készültek káosz alapú modellek [122]. A káosz vezérlésére az ún. OGY algoritmus [135] ismert, amelyet több dimenziós dinamikus esetben is sikeresen alkalmaztak gépészeti rendszerekben.

2.1.3 HIBRID MEGOLDÁSOK VEZÉRLÉSEKBEN

Sok erofeszítés történik, hogy a különféle MI megoldásokat ötvözzék komplex rendszerekben, és mindig azt a megoldást használják, amely az adott problémára a legmegfelelobb.

Az Intemor rendszer [144] neurális hálót, elore- és hátracsatolt következteto gépet, valamint eset alapú következtetést is használ komplex folyamatirányítási feladatra. A folyamatból érkezo jelek feldolgozásakor elobb az adatok kalibrálására neurális hálót, az állapot felügyeletre elorecsatolt következteto rendszert, az eseménynaplózáshoz eset alapú következtetést, majd a hiba diagnózishoz hátracsatolt következteto rendszert használ.

Tipikus ötvözésnek tekintheto, amikor szabály alapú rendszerekben egy közel optimális szabályhalmaz kiválasztása ill. generálása genetikus algoritmus alkalmazásával történik. Így rugalmas gyártórendszerben a köteg (batch) méretének kiválasztására szolgáló rendszert ismertetnek Deng és társai [50]. Folyamatos gyártás mellett a muhely szintu ütemtervet betartó, a megrendeléseket idoben eloállító és a gépek hatékony üzemeltetését biztosító kötegméretre van szükség, ahol a szerszámok rendelkezésre állását és a megmunkálási idoket is figyelembe kell venni. Egy másik alkalmazásban Egresits Csabáék fuzzy szabályok generálását és finomítását tanulással támogatják [54].

Neurális hálózat és szakérto rendszer együttes alkalmazása is gyakori. Az IntelliSPC rendszer [141], mely CIM rendszerek minoségbiztosítását támogatja SPC (statisztikus folyamatirányítás) felhasználásával. A folyamat mért jelei egy neurális háló alapú mintafelismerobe érkeznek, mely részleges mintákat is tud kezelni és kimenetén a folyamat státuszáról és kulcs paramétereirol ad értéket. Ezekkel az értékekkel dolgozik tovább egy szakérto rendszer, amely a minoség szempontjából veszélyes állapotokat képes felismerni. A rendszer kiegészül egy szimulátorral, ami a szakérto rendszer által felismert folyamat devianciáknak a minoségre vonatkoztatott költségeit számolja. Hasonló hibrid rendszer - bár az nem tartalmazott neurális hálót - Magyarországon is készült [5].

Gyártócellák menedzselésére, a belso üzemzavarok lekezelésére Kádár Botondék [93] reaktív tudásbázisú rendszert javasolnak, amely szakérto rendszerbol és neurális hálóból áll. A háló a mért jelekbol számított jellemzoket ad át a szakérto rendszernek, amely ezek mellett egy szerszám adatbázisban tárolt adatokat is felhasznál következtetéseihez.

A hibrid rendszerek mind a mai napig talán legnagyobb problémája, hogy az alkalmazott MI módszerek, szoftver eszközök integrációja legtöbbször az alkalmazás-fejleszto feladata marad. Léteznek ugyanakkor olyan keretrendszerek (pl. a G2 [65], amelyikben többféle (pl. szabályalapú, neurális és fuzzy) módszerek ötvözhetok hatékonyan, mert ezek - valamilyen szinten - eleve a rendszer részét képezik.

2.2 VEZÉRLÉSI FELADATOK MEGOLDÁSA INTELLIGENS ÁGENSEK SEGÍTSÉGÉVEL

Az MI területén a 90-es években, hasonlóan a számítástechnika általános fejlodéséhez, az elosztott rendszerek egyre inkább elotérbe kerültek. Az elosztott MI technológiának is a legfontosabb területe az ún. intelligens ágensek és alkalmazásuk. A 90-es években nagyon sokan próbálkoznak ágens alapon jó megoldásokat találni az intelligens gyártás kihívásaira. Shen és Norrie [154], valamint Parunak [137] összefoglaló cikkei alapján alapvetoen háromféle alkalmazási területre osztható fel, hogy mely területeken folynak ipari kutatások az ágens alapú rendszerek IMS-beli használatára:

- Szimuláció és modellezés a gyártási feladatok legkülönfélébb szintjén. Vannak olyan megközelítések, ahol az egyes ágenseknek pontosan meg van a fizikai megfeleltetésük (pl. eroforrások, munkadarabok, megrendelések stb.). Más esetekben fogalmak is lehetnek önálló ágensek (pl. célok, korlátozások stb.).

- A "gyártási piramis" tetején levo feladatok: vagyis a vállalat szintu számítógépes integráció (enterprise integration), a beszállítói lánc kezelése (supply chain management), humán felhasználók együttmuködését támogató (collaboration) megoldások, amelyekhez ma már feltétlenül hozzá kell venni az e-business, e-manufacturing, vagyis az elektromos üzleti és gyártási szoftver eszközöket.

- A hagyományosabbnak tekintheto sorrendtervezés, ütemezés és gyártásvezérlési feladatok, amelyek egyben az ágens alapú rendszerek gyakorlati alkalmazhatóságának az egyik tipikus kísérleti terepe. Ide sorolható az ún. HMS (holonikus gyártórendszer) koncepció [171], amely egy speciális probléma megközelítésnek tekintheto a gyártásautomatizálás témakörében. A konkrét alkalmazások azt bizonyítják, hogy alapvetoen itt is sorrendtervezési, ütemezési és gyártás vezérlési feladatok megoldására kerül sor. Cselényi József és Tóth Tibor holonikus termelo rendszerek logisztikai kérdéseit elemezték [47]. Márkus András és társai holonikus gyártásban a holonok kooperációjára egyfajta piaci modellt ismertetnek [109].

Az értekezés témakörébe szorosan ez utóbbi csoport részét képezo gyártásvezérlési feladatok tartoznak. Érdemes megállapítani, hogy vezérlési feladatokra lényegesen kevesebben (~20%) használták az ágenseket [154], mint az ütemezés támogatására. Az

alábbi lista azokat az ágens alapú kutatásokat, projekteket mutatja be, amelyek gyártásvezérlési feladatok megoldására is kísérletet tettek:

1. A MAS (Manufacturing Agility Server) ipari projekt [137] egy húsz állomásból álló hegeszto sor dinamikus terhelési viszonyainak kiegyenlítésére törekszik. Egy-egy munkadarabon (autókarosszéria) az állomásonként 4-4 hegeszto robot mintegy 800 hegesztést végez el összesen oly módon, hogy a munkadarab 30 másodpercig van egy állomáson. Az ütemezett hegesztési feladatok végrehajtása során sokféle zavar keletkezhet, amelyek mind a hegesztosor teljesítményeit rontják. A MAS rendszerben mind a robotok, mind a varratok ágensek, ha valamely robot ciklus ideje jóval nagyobb a többiénél, akkor a feladatainak egy részét (varratokat) átadja olyan robotnak, amelyhez más, de fizikailag közel eso varrat van rendelve. A döntést az ágensek közös megegyezéssel hozzák, amelyet végül az operátornak is jóvá kell hagynia. Látható, hogy az ágens alapú intelligens vezérlés logikailag a cella (hegeszto sor) gyárinformatikai szinten muködik.

2. A MetaMorph kísérleti architektúrán alapuló rendszer [180] logikai szintbe szervezett ágensek segítségével old meg sorrendtervezési, ütemezési és vezérlési feladatokat. Ezek az ágensszintek a következok: *vezérlés tervezési* (pl. megrendelések elosztása, termelési feladatok lebontása gyártási feladatokká, általános ütemezés), *végrehajtás vezérlési* (EC) és *vezérlés végrehajtási* (CE). Az EC ágensek végzik a real-time feladatok generálását, kiosztását, ellenorzik a futó CE ágensek muködését, lokálisan ütemezik a vezérlési feladatokat stb. Az ágensek belso struktúrája a következo modulokból áll: (1) a helyi vezérlés végrehajtó modul, (2) helyi vezérlés tervezés modul (meghozza a helyi döntéseket a helyi vezérlés viselkedéséről), (3) globális vezérlés koordináció modul (más ágensekkel egyezkedik), (4) kommunikációs interfész modul (más ágensekkel KQML [59], a vezérlési adatokat adott real-time protokoll szerint továbbítja). Ezzel a belso struktúrával rendelkezo ágensek egy elosztott real-time operációs rendszeren futnak. A rendszernek csak a koncepciót igazoló prototípusa ismert.

3. A HMS koncepció keretén belül a vezérlés kérdéseit is vizsgálták [40], egy Petri hálón alapuló kísérleti rendszerben részletesen elemezték [39]. Az eroforrásokat és a megrendeléseket lokális ágenseknek (holonoknak), az ütemezot pedig központinak definiálva a következo holonikus vezérlési koncepciókat dolgozta ki Luc Bongaerts (Leuven):

    - *Hierarchikus elosztott kontroll*, amely azt jelenti, hogy mindegyik lokális holon rendelkezik saját kontrollal, azonnal reagálnak a zavarokra, egymással kommunikálnak, ugyanakkor a központi, az ütemezo holon az egész rendszer figyelembevételével küld információt a többinek. Ez a struktúra lehetoséget teremt a teljes rendszer teljesítményének optimalizálására, valamint növeli a rendszer jövobeli viselkedésének eloreláthatóságát.

    - *Elosztott döntéshozatal* a lokális és központi holonok között egy redundanciát jelent, de az autonóm lokális holonokat központi tanácsokkal tudja támogatni.

    - A *konkurens ütemezés és ütemterv végrehajtás* azonnali választ biztosít a rendszerbe fellépo olyan zavarok esetében, amelyek miatt az eredeti ütemterv felborul. Ehhez párhuzamosan egy vezérlo és egy ütemterv

optimalizáló ágens fut folyamatosan. Zavar esetén a vezérlo ágens azonnal reagál, bár az nem az optimális válasz lesz, majd amikor - egy ido után - az optimalizáló ágens is elkészül a válaszával kiértékelve a megváltozott helyzetet és egy új ütemtervet eloállítva (reaktív ütemezés [165]), ezt veszi át a vezérlo ágens. Ezek az ágensek mind a lokális, mind a központi holonban futnak.

Ez a HMS-beli megoldás is elsosorban cella szintu ágens alapú intelligens vezérlést valósít meg, a holonok segítségével realizált gyors reaktív ütemezés segítségével. Hasonló a helyzet a téma magyar kutatásaiban is [119].

4. Szintén alacsony színtu vezérlések holonikus szervezésének a kísérleti példája az a rendszer [179], ahol a teljes rendszer hatékony muködéséhez az eroforrásokat (vezérloket) dinamikusan újrakonfigurálják, dinamikusan átcsoportosítják az állandóan változó feladatok változó követelményeinek a hatékonyabb kiszolgálása céljából. Technikailag ez az újonnan fellépo feladatokhoz létrehozott ún. *dinamikus közvetíto* holon (DMH) segítségével történik, mely az adott feladatra összefogja az alacsonyabb szintu holonokat. Ez a rendszer gyakorlatilag a korábban ismertetett MetaMorph architektúra holonikus változata.

5. Talán a legkomolyabb ágens alapú vezérlo fejlesztés, amely a berendezés szintjén kísérel meg intelligens vezérlést nyújtani a Vanderbilt Egyetem intelligens robot vezérloje (IMA) [136], amely sok más modern robotvezérlési elvet igyekszik ötvözni az ágens technológiával. Az intelligens robotvezérlo célja, hogy dinamikusan allokálja egy adott problémához a "legjobb elérheto" megoldást. A rendszer muködésének lényegi elve az ún. akció kiválasztási mechanizmus, vagyis annak a menedzselése, hogy többféle muködteto és érzékelo mechanizmus közül lehessen rugalmasan választani egy konkrét szituációban, és a kiválasztottal átkonfigurálni a vezérlést. Az IMA különféle ágensekbol épül fel. A rendszer fizikai interfészeihez a hajtás ágensek és az érzékelo ágensek csatlakoznak. Az egyszerubb feladatokat az ún. viselkedés ágensek realizálják (pl. ütközés elkerülés, végállás figyelés), míg a legbonyolultabb összetett feladatokat (tárgy megfogása, navigálás)az ún. képesség ágensek. A multiágens környezetben egy adott feladat több lehetséges megoldása közül a feladat ágensek szavazásai nyomán alakul ki, hogy melyik megoldást hajtja végre a rendszer. Az IMA architektúra segítségével egy kétkarú, látószenzorral ellátott humanoid robot (ISAC III) és egy mozgó manipulátor (Helpmate) vezérlését oldották meg sikerrel.

6. Rzevsky modelljében [147], mely inkább egy funkcionális dekompozíciónak tekintheto, 5 intelligens ágens épít fel egy szerszámgépet. Az elso az anyag leválasztási sebességét optimalizálja a különféle munkadarabokra. A második a megmunkálógép állapotát felügyeli a leheto legjobb megmunkálási körülményeket biztosítva. Egy esetleges szerszámtörést ez detektál és lassítja a sebességet az elso ágenssel konfrontálódva. A harmadik ágens ütemezi a munkavállalást, amellyel igyekszik minimalizálni a szerszámgép üresjáratát. A negyedik összegyujti a gép összes tevékenységének és kötelezettségeinek az adatait és - ha szükséges - figyelmezteto üzenetet küld a többi ágensnek. Az ötödik felügyeli a gép közvetlen környezetét, felel a kiszolgáló berendezésekkel való ütközés elhárításáért. Hagyományosabb (tengely, szerszám, munkadarab, befogó stb.), de ugyancsak funkcionális ágens felosztást használ a francia Shiva rendszer [138].

Fontos megállapítani, hogy az intelligens ágens alapú vezérlésre (is) használt rendszerekben az ágensek tipikusan nem csak az egyes berendezéseket reprezentálják, hanem annál kisebb egységeket is. A holonikus gyártórendszer egyfajta referencia architektúrájában (PROSA) önálló holonként jelennek meg a megrendelések, a termékek és az eroforrások [172].

2.3 G2 - INTELLIGENS ALKALMAZÁSOK SZOFTVER KÖRNYEZETE

Az értekezésben több helyen szerepel a Gensym G2 nevu szoftvere [65], mint az egyes kutatási eredmények realizálásának, a különféle prototípusok megalkotásának az eszköze. A G2 egy olyan fejleszto és futtató környezet, amely intelligens alkalmazások létrehozását támogatja, és a fejlesztés teljes életciklusában használható. A következo alapveto tulajdonságokkal rendelkezik:

- A beépített objektum orientált szoftver elemkészlettel rendelkezo rendszer objektumaihoz bármely általános objektumosztály szinten hozzárendelheto bármilyen és bármennyi ún. attribútum, valamint szekvenciális végrehajtással rendelkezo ún. metódus is.

- Következteto gépe többféle szabályt és következtetési mechanizmust ismer (esemény és adatvezérelt), a szabályok érvényességi köre megadható.

- Támogatja a szabály és modell alapú következtetést is, valamint az eljárásokban kódolt adatfeldolgozást, ezeket párhuzamosan képes futtatni (eljárások és szabályhalmazok), így egyszerre több feladat kiszolgálására is alkalmas.

- A grafikai építoelemek nemcsak vizuálisan jelenítik meg az objektumokat, hanem a velük végzett grafikai muveletek (pl. összekapcsolás) programozást is jelentenek. Strukturált, a természetes (angol) nyelvhez közeli programozói szintaxis támogatja a fejleszto és a végfelhasználó közötti párbeszédet. Vagyis - igény esetén - a végfelhasználók is némi energia befektetéssel jól értik és tudják olvasni a G2-ben írt programelemeket.

- A felhasználó által létrehozott osztályok, objektumok szervesen beépülnek futási idoben a rendszer elemkészletébe, menürendszereibe. Objektum könyvtárak és funkció modulok segítségével inkrementálisan lehet programot fejleszteni. Így a program kis részének elkészülte után már az adott rész futtatható, tesztelheto.

- Többféle beépített teszt eszköz áll rendelkezésre (pl. dinamikus szimulátor, a rendszer teljesítoképességének on-line vizsgálata). Tudásbázisába a modulok futási idoben tölthetok be és törölhetok, minden eleme módosítható.

- Nagy sebességu és sokszorosan bevált kommunikációs eszközt biztosít adatbázisokkal, SCADA rendszerekkel, terepi buszokkal és más szoftver rendszerekkel való együttmuködésre, közvetlen ActiveX, Java, C/C++ és Corba kapcsolattal rendelkezik.

- Opcionálisan más mesterséges intelligencia módszer (neurális háló, fuzzy stb.) moduljával bovítheto.

3 OSZTÁLYOZÁSI MODELL INTELLIGENS GYÁRTÓRENDSZEREK KOMMUNIKÁCIÓJÁRA

Elosztott gyártórendszerekben az üzenetek modellezésére és analízisére sokféle megoldás lehetséges, Lin és Zhou [105] például Petri hálózatot használ, bár náluk nem teljesen világos kutatásuk gyártórendszer specifikussága.

Az elemzések elsosorban két területet vizsgálnak: az üzenetek mennyiségét és a válaszidoket. Az elobbi azt jelenti, hogy az adott eszközök, rendszerek között mekkora az adatforgalom, ennek milyen dinamikus tulajdonságai vannak, pl. hol és mi okoz szuk adatátviteli keresztmetszetet a gyártórendszerben. A válaszidok betarthatósága pedig mindig az irányíthatóság szempontjából kritikus. A szakirodalom hagyományos megállapítása, hogy a közismert "piramis" gyárinformatika modell esetében alul gyors válaszidok és kisebb adatmennyiség, magasabb szinten nagyságrendekkel nagyobb idok és adatmennyiségek vannak (pl. [57]).

Ez az elv megdolni látszik napjainkban, elsosorban a megmunkálási folyamatok részletesebb megfigyelése és felügyelete miatt a megmunkálás közeli szinteken is óriási adatmennyiség jelenik meg. Ennek oka a különféle fizikai jelek (rezgés, ero, akusztóemisszió stb.) mérése szenzorokkal, és ezeknek a jeleknek az ido és frekvenciatartományban való jelfeldolgozása, majd kiértékelése (pl. valamilyen szubszimbolikus MI módszerrel). Emellett a szabad szemmel, a megmunkáló gép mellol nem jól megfigyelheto vizsgálatokhoz teret nyer a videó használata, aminek a segítségével a kezelo lehetoségei ugrásszeruen megnonek a megmunkálás felügyeletére. Ugyanakkor ez egyre kisebb problémát jelent, hiszen az informatika fejlodésével egyre csökken a berendezések, rendszerek közötti átviteli ido és no az egységnyi átviheto adatmennyiség, amit teljesen hétköznapi kereskedelmi eszközökkel is könnyedén garantálni lehet (pl. ma már egy 100 Mbit/sec-os adathálózati interfész bármelyik PC-nek elemi tartozéka).

Az intelligens gyártórendszerek esetében a berendezések közötti kommunikáció alaposabb elemzésére új osztályozási szempontokat és kategóriákat javaslok értelmezni figyelembe véve az elosztott MI fogalmait.

3.1 AZ INTELLIGENS KOMMUNIKÁCIÓ LOGIKAI SZINTJEI

Gyártórendszerekben - de más hasonló autonóm számítási elemekbol álló komplex rendszerekben is - vizsgálható, hogy az egyes eszközök intelligens viselkedése milyen módon látható, érzékelheto a közöttük levo kommunikáció alapján. Elosztott intelligencia esetében várható, hogy az üzenetek tartalmából képet lehet kapni arra, hogy az eszközök egymás közötti kommunikációja milyen kapcsolatban áll a készülékekben futó MI megoldásokkal.

Bevezetem a logikai kommunikációs szint fogalmát a gyártórendszerben, ami a gyártóberendezések közötti egyedi üzeneteket sorolja szintekbe azok tartalma alapján. Ez a gyakorlati kategorizálás a rendszerben levo intelligencia elosztottságát teszi

vizsgálhatóvá, valamint annak milyenségére is utal. Az alábbi három kategóriát különítem el [24]: (1) vezérlési adat; (2) tudás-gyujto; (3) ismeret-megosztó:

1. A vezérlési adat szintet, amely a leggyakoribb, az olyan üzenetek jelentik, ahol az egyes készülékek közötti kommunikáció legfeljebb a különféle adatgyujtéseket és vezérlési parancsokat foglalja magában. A berendezések közötti kommunikáció semmilyen módon nem utal arra, hogy az egyes készülékekben MI alapú megoldások is jelen vannak, a berendezések közötti kommunikáció teljesen megegyezik az egyéb, nem MI alapú rendszerekkel. Három alesetet tudok megkülönböztetni:

   - Egyáltalán nincs MI eljárás a rendszerben.

   - Az MI eljárások teljes mértékben lokálisan muködnek, nincs semmilyen egymásra hatásuk, bár lehet, hogy több MI alkalmazás fut a rendszerben.

   - Több MI alkalmazás fut, ezek használják is egymás eredményét - vagyis valódi elosztott MI rendszer van - de ennek semmilyen nyoma nincs a kommunikációban.
     (Szemléletes példa lehet erre az esetre a bridzs kártyajátékban a licit. Formálisan semmi nem mutat arra, hogy egyéb tartalma is van az üzenetnek, ill. a játék keretei nem engedik meg, hogy egyéb üzeneteket is váltsanak a partnerek.)
     Ez utóbbi fordulhat elo gyártórendszerekben, amikor a meglévo eszközök kommunikációjának a korlátai nem teszik lehetové az intelligens üzenetek továbbítását.

2. A lokálisan használt MI megoldások használata esetén lehetséges, hogy az egyes MI algoritmusok következtetéseikhez, ill. esetleges öntanulásukhoz valamilyen - más berendezésekben tárolt - adatokat gyujtenek, amit tudásgyujtésnek (knowledge acquisition) neveznek hagyományosan, és ez nálam az intelligens kommunikációnak már egy magasabb szintjét jelenti. A tudásgyujtés során szokás megkülönböztetni azt a két esetet, amikor egyszeruen valahol már meglévo adatokat összegyujtve történik a tudás megszerzése (passzív), vagy amikor a tudásgyujtés aktív, tehát speciális (pl. diagnosztikai) eljárások futtatása történik egy másik berendezésben, és ennek adatait kapja meg az "intelligens" berendezés.

3. Az intelligens kommunikáció legmagasabb szintje felfogásom szerint az, ahol az intelligens készülékek között már ismeret/tudás megosztás (knowledge communication/sharing) történik. Az egyes "intelligens" berendezések a használt MI megoldásuktól függoen kérdeznek egymástól, intelligens ágensekként támogatják egymás muködését. Tanuló rendszer esetén egy frissen felismert összefüggés, szabály elterjesztése is idetartozik.

Egy adott intelligens gyártórendszerben a háromféle intelligens kommunikációs szintnek megfelelo üzenetek áramlanak a berendezések között, és összekeveredve az adott kommunikációs nyelv, környezet kívánalmai szerint. Hangsúlyozom, hogy az üzeneteket tartalmilag kell vizsgálni, hogy melyik logikai szinthez tartozók, a kerete egyiket sem sorolja be automatikusan valahová. Így pl. egy ágens kommunikációs

nyelven átadott üzenet (3.1.2 alfejezet) önmagában nem garantálja, hogy az üzenet tartalma ténylegesen tudásgyujto vagy -megosztó-e.

A három szint egyfajta hierarchiát is meghatároz, hiszen egy magasabb szint meglétéhez szükséges az alacsonyabb is. Az iparban üzemszeruen muködo gyártórendszerekre túlnyomó részben az elso szint kizárólagossága a jellemzo, néhány helyen a második is megtalálható. Kutatási prototípusokban lehet ismeretmegosztást is támogató elosztott MI vezérlésekkel találkozni. Az 1. ábra a logikailag szétválasztott háromszintu intelligens kommunikációs hierarchiát mutatja be. (Az ábrán néhány készülék intelligens, a többi nem.)

1. ábra Az intelligens kommunikáció logikai szintjei gyártórendszerekben [9]

A továbbiakban megmutatom, hogy ez az általános osztályozás magában foglalja mind az alkalmazott ontológia (3.1.1 alfejezet) megoldás - amely az üzenet értelmezés módjáról árulkodik -, mind a szükséges kommunikációs protokoll - amely az üzenet továbbításának a keretét adja - ismert felosztásait.

3.1.1 TUDÁSMEGOSZTÁSI KATEGÓRIÁK

A tudás megosztásának elengedhetetlen feltétele, hogy a kommunikációban résztvevok "egy nyelvet" beszéljenek. Ennek a kérdésnek a vizsgálata az ún. ontológiák meglétét és azok kapcsolódását jelenti. Az ontológia egy fogalomalkotás világos, pontos leírása [70]; maga az elnevezés a filozófiából lett átvéve. Tudásalapú rendszerekben bármilyen elem, amit a rendszer használ, vagyis "létezik", az valamilyen módon reprezentálható. A konkrét ontológia adja meg, hogyan kell konzisztensen az adott komplex környezetben a különbözo fogalmakat, elemeket ábrázolni. A konzisztencia szükséges, ha az elemek kapcsolatát, az általuk reprezentált tudást meg akarjuk osztani, át akarjuk adni rendszereink között.

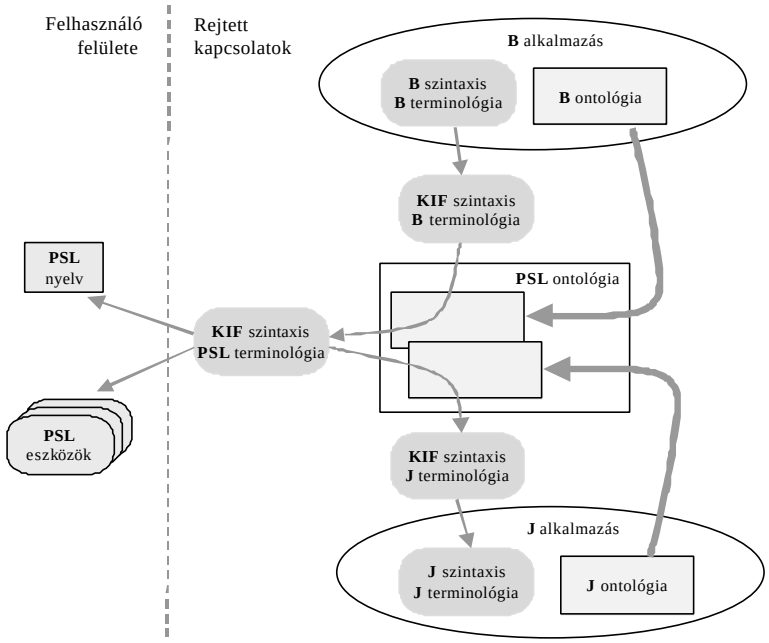
A tudás megosztást gátolja, hogy az információ jelentése nagymértékben függ a környezettol, amelyben az információ keletkezik ill., értelmezodik. A gyártórendszerekben ez fokozottan igaz, hisz egyre no azok komplexitása, a bennük használt szoftver eszközök száma és célja, és a feldolgozott adatok és információk

mennyisége, jellege. Az emberi kommunikációban is pl. a tervezésben, a gyártásban és az értékesítésben más és más terminológiát használnak. Ez a kooperáló intelligens eszközök esetében fokozottan érvényes.

A megoldást a gyártáshoz használt közös ontológia alkalmazása jelenti, ahol persze problémás lehet annak bonyolultsága, esetleges életidegensége (a humán felhasználók számára túl absztrakt vagy nehézkes). Ilyen fejlesztés több helyen folyik, pl. a SHADE projekt a Stanfordon [112], a PSL projekt a NIST-ben [95], ESPRIT kutatások [140], de értelemszeruen ide sorolhatók a STEP szabvány körüli kutatások is. A NewSun [42] környezet is alkalmas arra, hogy már korábban elkészült tudás modelleket, ágenseket kapcsoljon össze CORBA alapon komplex tervezési feladatok intelligens támogatására. Tudásbázisok kooperálásának a kutatása során ugyancsak a Stanford Egyetemen definiálták és fejlesztették ki az OKBC (Open Knowledge Base Connectivity) API-t, amely alkalmas különbözo módon reprezentált tudás egységes elérésére [43].

Véleményem szerint hibás az az elképzelés, amely egyetlen reprezentációt (ontológiát) kíván használni a teljes rendszerben, hiszen - ismerve a jelenlegi sokféle megoldást - bármely ontológia teljes köru ipari elterjedése elképzelhetetlen. Azok a megoldások, ahol az egyik ágens kimenete jelenti a másik bemenetét, de nem használják egymás beépített tudásmodelljét nem elonyösek a legtöbb esetben, hisz jelentosen lecsökken a kooperáció lehetosége.

Nem marad más, mint a meglévo belso reprezentációkat minden kommunikálni akaró rendszerben, ágensben, (melyek akár kimondatlanul is megvannak,) egy közös ontológia szerint át kell konvertálni, hogy lehetséges legyen a tudás megosztása. A 2. ábra a PSL-ben elképzelt megoldást mutatja, mely a Stanford KIF [62] szintaxisát használja:



2. ábra Független tudásalapú alkalmazások kommunikációja közös PSL ontológia segítségével [95]

Ez azt jelenti, hogy az ontológiák körében az alábbi szintek különböztethetok meg:

0. Nincs explicit ontológia (pl. az alkalmazott MI módszer nem szimbolikus, ill. nem használ szimbolikus reprezentációt)
1. Zárt ontológiák, vagyis az egyes rendszerek tudásábrázolása on-line, nem osztható meg más rendszerekkel.
2. Egyedi ontológiák, ahol speciális procedurális interfészeken keresztül van lehetoségük az egyes rendszereknek, hogy egymás szolgáltatásait elérjék, de ontológia konverzióra nincs lehetoség.
3. Összekapcsolható ontológiák, melyek képesek egymással a bennük meglevo tudás megosztására akár azért, mert ugyanazt az ontológiát használják, akár azért, mert az ontológiáik átkonvertálhatók egymásba.

A 2. táblázat mutatja, hogy a lényegében egyértelmu a megfeleltetés, amit találtam, az ontológiák felosztása és az általam definiált logikai kommunikáció osztályozása között.

	zárt	**egyedi**	**összekapcsolható**
vezérlési adatok	+	felesleges	felesleges
tudásgyujto	nehézkes	+	felesleges
ismeret megosztó	nem lehetséges	nehézkes	+

2. táblázat Logikai szintek és ontológiák kapcsolata gyártórendszerekben

3.1.2 INTELLIGENS RENDSZEREK KOMMUNIKÁCIÓS PROTOKOLLJAI

A gyártórendszerek világában nagyon sokféle protokoll ismert a különbözo feladatokra, sajnos sokkal több, mint amennyi szükséges lenne. A gyárinformatika alsó szintjein a 80-as években el lehetett mondani, hogy ahány jelentos gyártó, annyiféle protokoll volt használatban. Mára a helyzet sokat javult, de a lényegében ugyanarra a funkcióra kidolgozott (de-facto) szabványok száma még mindig túl sok. A nyílt rendszerekkel kapcsolatban (4. fejezet) többféle protokoll is szóba kerül majd. Ebben az alfejezetben az az érdekes, hogy milyen keretet biztosít a protokoll a gyártórendszer egyes elemei közötti kommunikációhoz.

Az üzenetekben az egyes felek lehetnek aktívak, passzívak; lehet közöttük alá/fölérendelt viszony (master/slave) vagy sem (peer to peer). Ezek a kategóriák nem az üzenetek tartalmi keretéhez kapcsolódnak, tipikusan alacsonyabb szintu protokollokban realizálódnak.

1. Hagyományosan a CNC-k és a robotvezérlések meglehetosen zártak voltak. A gyártóspecifikus DNC csatornájukon keresztül minimális, és szigorúan rögzített (elore specifikált) adatokat lehetett lekérdezni, beállítani. Lényegében ezzel a megoldással rokon, ha távoli eljáráshívással (RPC) lehet a berendezéssel kommunikálni.

2. A 90-es években terjedtek el a különféle üzenetalapú megoldások, amelyekben a protokollok mind rokoníthatók az objektum orientált metodikával (pl. MMS), vagy eleve arra épülnek (pl. CORBA). Ezeknél a megoldásoknál a berendezések, mint objektumok, virtuális gépek „látszódnak” a hálózat felol.

3. Kutatási szinten jelen van a gyárinformatikában az ágens alapú kommunikáció, amely az emberi beszédbol intuíciókat merítve, alakítja ki az egyes ágensek közötti párbeszédet (szólásaktus – speech act). Az egyik legismertebb ágens kommunikációs nyelv (ACL) a KQML (Knowledge Query and Manipulation Language) [59].

Mindhárom esetben az üzenet jellegének az alaptípusai: állító vagy kéro. A szólásaktus esetében ez kiegészül sok egyébbel (pl. elfogadás, engedélykérés, szavazat, javaslat, visszautasítás, ajánlat stb. [175]).

Az 3. táblázatban mutatom be a protokollok és a logikai szintek kapcsolatát. Itt nem olyan markáns a megfeleltetés, mint az ontológiáknál (3.1.1 alfejezet). Szerintem ennek oka az, hogy egy adott protokoll szerepe, hogy egy keretet adjon az üzenetek számára és nyilván lehetséges egy "kényelmetlenebb" keretben is megvalósítani egy adott logikai szintu párbeszédet.

	procedurális	**üzenet alapú (objektum orientált)**	**szólásaktus (ágens kommunikáció)**
vezérlési adatok	+	+	felesleges
tudásgyujto	+	+	+
ismeret megosztó	nem lehetséges	nehézkes	+

3. táblázat Logikai szintek és protokoll fajták kapcsolata gyártórendszerekben

Többek szerint (pl. Kaula [92]) az objektum orientált megközelítés azért nehézkesebb az ismeret megosztási feladatokban, mert nem tartalmaz az üzenet feldolgozására nézve semmilyen járulékos információt, szemben a szólásaktusos protokollokkal.

3.2 GYÁRTÓRENDSZEREK VIZSGÁLATA A KOMMUNIKÁCIÓ LOGIKAI SZINTJEI ALAPJÁN

A logikai kommunikációs szint lehetoséget ad arra, hogy vizsgálni lehessen az intelligencia elosztottságát egy muködo gyártórendszerben az üzenetek alapján.

A gyártórendszerben zajló kommunikáció tervezése Gullander és társai szerint kulcsfontosságú a vezérlés struktúrájának a kialakításánál [72]. A logikai szintek bevezetésével lehetoség nyílik különféle gyártórendszerek hálózati kommunikációs forgalmának mennyiségi és az intelligencia elosztottságának vizsgálatára. Kérdés, megállapítható-e, hogy futás közben milyen mértékben kooperálnak az egyes berendezések, milyen tudásszerzo tevékenységet folytatnak.

Látható, hogy a logikai szintek fogalma a megfigyelhetoség kérdését veti fel, és ezzel kapcsolatban a következo problémákat: (1) eldöntheto-e egy üzenetrol, hogy melyik csoportba tartozik; (2) van-e jelentosége annak, hogy milyen a háromféle üzenet aránya egy gyártórendszer adott idoszakában?

1. Az elso kérdésre a válasz biztosan igen, hiszen a második és harmadik szintet képezo üzenetek a rendszer alapmuködésén túli üzeneteket reprezentálják, így kiszurhetoek. Közöttük pedig az üzenet jellege nyomán (információ gyujtés vagy megosztás) lehet különbséget tenni. Természetesen az egyes készülékek ismerete (pl. az X. berendezés csak elso szintu üzenetet küld) is komoly támpontot jelent.

2. A második kérdésre a válasz pedig az, hogy nem az arány érdekes, hanem ennek változása a gyártórendszerben történt események tükrében.

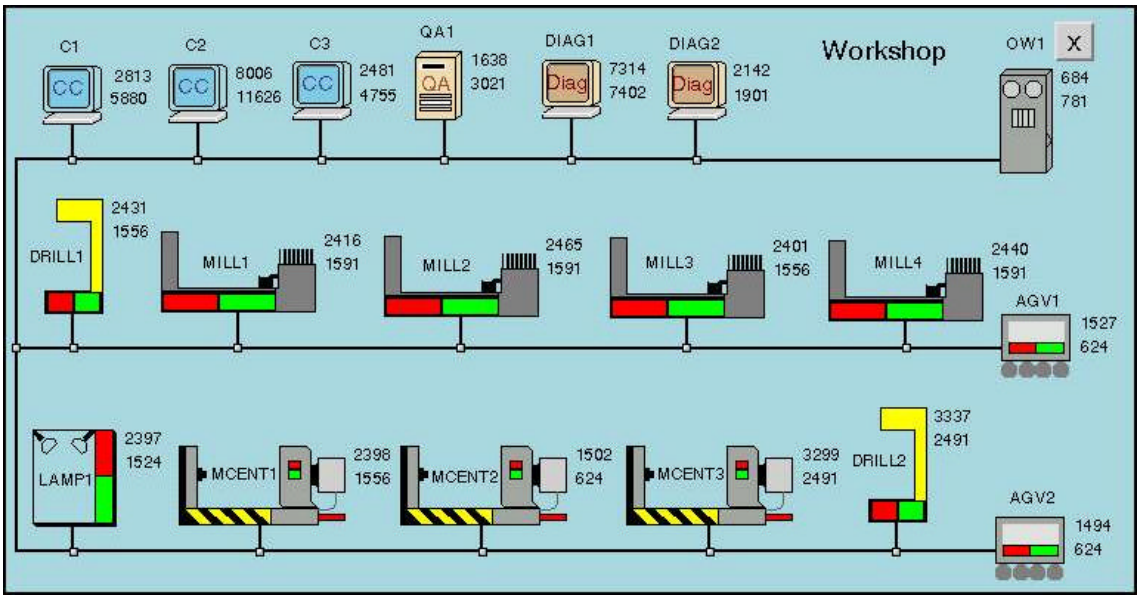
A dolgot bonyolítja, hogy nyilvánvalóan a mérheto adatok mennyisége, fajtája, jellegzetességei mind különbözoek, és függenek a konkrét gyártórendszertol és annak kommunikációs kapcsolataitól. A nehézségek ellenére kétféle elemzésre nyílik lehetoség:

- A logikai szintekbe sorolt üzenetek összegzése. Ez lehetoséget teremt a gyártórendszer "normál" muködésének és a tervezés során feltételezett üzemelési modelljének az összehasonlítására, így a - sokszor kritikával kezelt - MI alapú módszerek valódi muködésének egyfajta - az egész gyártórendszerre való hatását figyelembevevo - vizsgálatára.

- A problémamentes muködés és valamilyen zavar, deviancia hatásának összehasonlító vizsgálata. Ez egy berendezés kikapcsolását vagy meghibásodását, valami miatt a gyártási ütemterv váratlan megváltozását, a gyártás minoségében fellépo zavarokat stb. jelentheti. Izgalmas kérdést jelent ilyenkor, hogy vajon ezek a zavarok milyen hatással vannak a különbözo üzenet típusokra, vagyis "aktívabbak-e" az intelligens funkciók.

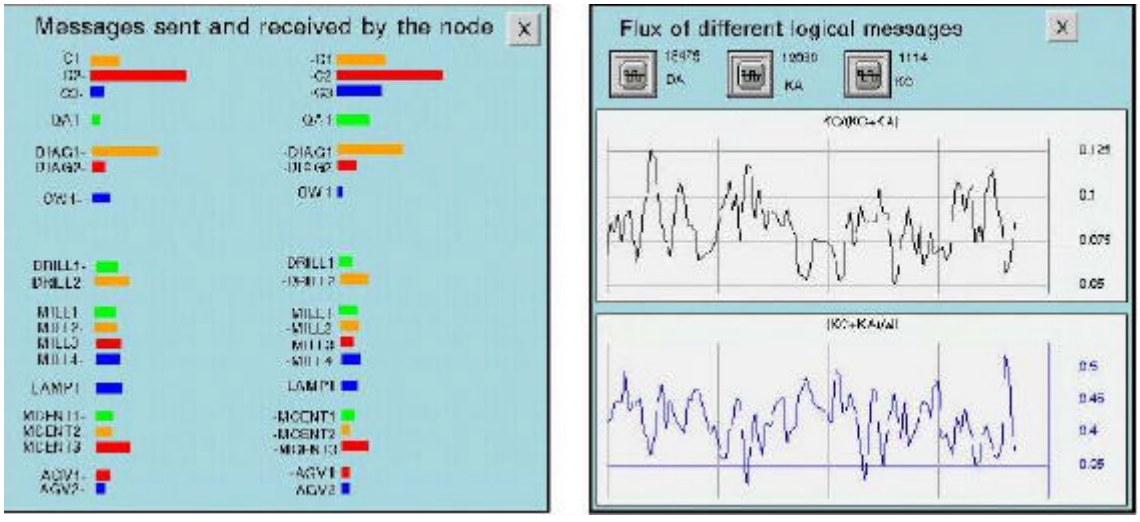
A fenti elvi vizsgálati módszer gyakorlati kipróbálására készült el egy analizátor-szimulátor [20] a G2 rendszerben (2.3 alfejezet), amely a SzTAKI-ban korábban kifejlesztett gyártórendszer szimulátor [6] tapasztalatait is felhasználja. A szimulátor lehetové teszi egy gyártórendszer megmunkáló és irányító elemeinek a felvételét, a közöttük levo hierarchikus kapcsolatok definiálását (pl. melyik cellavezérlohöz melyik megmunkáló gép tartozik), valamint annak meghatározását, hogy mely berendezések intelligensek és ezek hogyan kommunikálnak a környezetükkel. A konfigurálás után elindítva, a rendszer szimulált hálózati forgalmat generál. Muködés közben nem várt események bekövetkezése is megadható. Az analizátor rész gyujti az eszközök közötti üzeneteket, egy körpufferje segítségével az utolsó 500 üzenetbol pillanatnyi statisztikák is számíthatók. Elméletileg a G2 külso interfészén keresztül egy valódi ipari hálózatra is rákapcsolható lenne.

A 3. ábra konfigurációjában 3 megmunkálógép (*MCENTx*), 3 marógép (*MILLx*), 2 fúrógép (*DRILLx*) és 1 mérogép (*LAMP1*) alkotja a muhelyt, amelyeket két szellemkocsi (*AGVx*) szolgál ki. A gyártórendszert 3 cellavezérlo (*Cx*), 1 minoségbiztosítást felügyelo (*QAx*) és 2 diagnosztikai (*DIAGx*) számítógép muködteti. A külvilággal való egyéb kapcsolatot formálisan egy további állomás (*OWx*) helyettesíti a modellben. Az ábrán az egyes állomásokból induló és hozzájuk érkezo üzenetek száma is látható a teljes vizsgált idotartomány alatt.



3. ábra Az intelligens kommunikációs szintek vizsgálatához használt szimulált gyártórendszer

A 4. ábra mutatja azokat az ablakokat, amelyek a pillanatnyi kiértékeléseket (utolsó *n* darab üzenet) jelenítik meg. A baloldalon látható, hogy melyik géptol hány üzenet indult és érkezett; a jobb oldalon alul a második és harmadik szintu (intelligens) üzenetek aránya az összes üzenethez viszonyítva; felül a harmadik szintu üzenetek aránya az összes intelligens üzenet függvényében.



4. ábra Pillanatnyi üzenetek kiértékelési lehetoségei

4 NYÍLT IPARI VEZÉRLÉSEK

A nyolcvanas évek vége óta egyre nagyobb igény a szerszámgépgyártók és a felhasználók részérol, hogy a CNC, PLC és robotvezérlo berendezések se legyenek zártak. Lehetoség legyen a meglévo eszközhöz egyedi szoftverfejlesztéssel új funkciókat minél egyszerubben hozzákapcsolni, adott funkciókat más szoftverekkel helyettesíteni, az alkalmazásokat "testre szabni" stb. Fogalmilag csak részben tartozik ide és helytelen szóhasználat - sokszor nem több, mint piaci szlogen -, hogy az olyan vezérléseket is nyíltnak nevezik, amelyek hardvere PC. Tény, hogy a PC hardver és különösen a felette futó MS WinNT vagy MS Win98 operációs rendszer sok olyan lehetoséget biztosít, amelyre jellemzo egyfajta nyíltság, de önmagában a platform nem tesz nyílttá egyetlen vezérlést sem.

Az sem minden esetben egyértelmu, hogy mit kell érteni pontosan a nyílt vezérlo fogalmán. A nyílt rendszerekre kiindulópontként az IEEE informatikai definícióját használom: *Egy nyílt rendszer olyan képességeket biztosít, amelyek lehetové teszik helyesen implementált alkalmazások számára azt, hogy különbözo gyártóktól származó különféle platformokon fussanak, más rendszeralkalmazásokkal együtt tudjon muködni és a felhasználók számára egységes stílusú interakciókat nyújtson* [1]. Gépvezérlések esetében ez a definíció 5 féle nyíltsági kritériumra bontható, amely a felhasználói igények felol közelíti meg a kérdést:

1	Nyílt és moduláris rendszer architektúra, amely tartalmaz egy referencia modellt,	e nélkül bármilyen API definiálása csak fél munka, mert az egyes implementációk továbbra is ad-hoc módon fognak kialakulni.
2	Nyílt interfész a modulok közötti kommunikáció számára,	amely független attól, hogy a vezérlés elosztott vagy sem.
3	Nyílt és szabványos felhasználói felület,	amely biztosítja, hogy a kezelo a szokásos funkciókat gyártótól függetlenül ugyanúgy érhesse el, használhassa, és a vezérlo képernyojén ugyanúgy lássa.
4	Nyílt és szabványos felület a külso kapcsolatok számára	(a hajtások, beavatkozók és a vezérlés között, valamint a vezérlés és a más egységek (többi vezérlo, PLC, cellavezérlo stb.) között.
5	Szabványos NC programnyelv,	amely biztosítja, hogy egy adott munkadarab gyártási programleírása CAD rendszertol, off-line programozói munkahelytol, vezérléstol és megmunkálógéptol független legyen. Hasonlóan egy adott muveletsort leíró robotprogram is független legyen a robottól és attól, hogy melyik rendszer generálta.

4. táblázat Nyílt vezérlés kritériumai

Az egyes kutatásokban és termékekben ezek a területek változó súllyal jelentkeznek. Szukebb értelemben csak a 4. táblázat 1. és 2. pontjával szokás foglalkozni, a következo (4.1) alfejezetben az áttekintés ezek kérdéseit részletesen is tárgyalja.

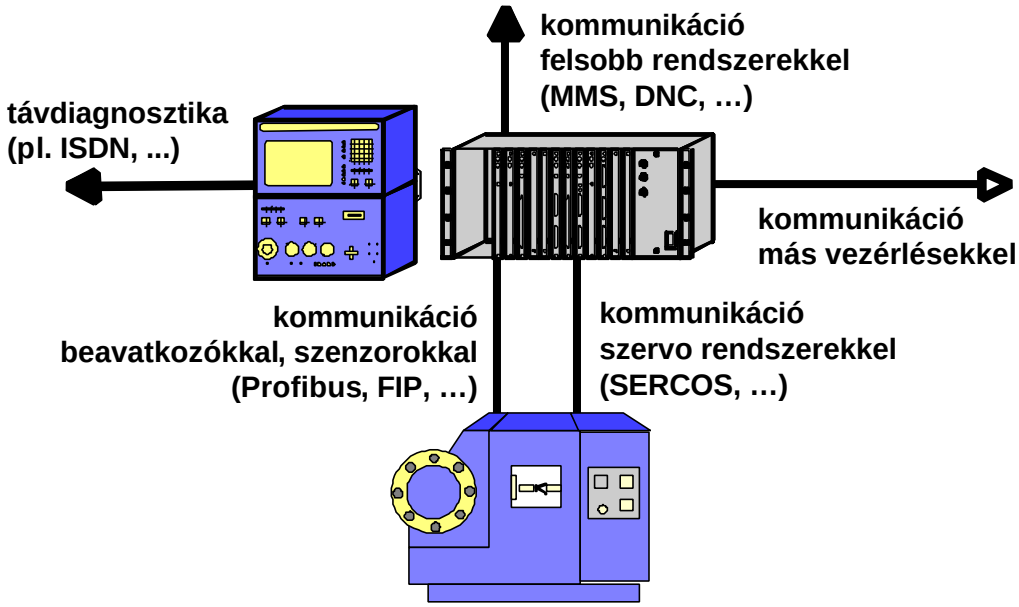
A 4. táblázat 3. pontjával kapcsolatban megállapítható, hogy a legtöbb rendszer számára a nyílt felhasználói interfész (HMI) egyet jelent a hagyományos MS Windows felülettel, ahol a rendszerintegrátornak, szerszámgépgyártónak meg van az a lehetosége, hogy maga alakítsa ki az adott berendezés egyedi felhasználói felületét. Ezzel szemben

a nagy végfelhasználók egységes, gyártó független felhasználói felületet akarnak, meghagyva a lehetoséget egyedi ablakok hozzáépítésére. Ilyen HMI az OSACA rendszerben (4.2.1 alfejezet) lett részletesen specifikálva [132], az OMAC-ban (4.2.3 alfejezet) a meglévo kutatási eredményeket és a létezo termékeket próbálják integrálni.

A vezérlés külso kapcsolatai (a 4. táblázat 4. pontja) sokfélék, több szabvány (ipari és de-facto) ismert és használatos. Az elmúlt évtized változása, hogy mind az alsó, mind a felso szintu kommunikációban megjelentek a hálózatok.

A felso szinten az általános LAN technológia (Ethernet, TCP/IP) vált meghatározóvá, az alsó szinten a digitális hajtások és a terepi buszok. A felso szintu és a belso modulok közötti kommunikációs kapcsolat kérdései némileg összemosódtak az elosztott vezérlések elotérbe kerülésével. Törekvések vannak (pl. Erdélyi Ferenc [57]) a felsobb szinten is modul architektúra kialakítására.

A digitális hajtások, a Siemens és Fanuc egyedi megoldásai mellett három fontosabb nyílt megoldás [152] ismert: a SERCOS (Seriell Realtime Communication System), a MACRO (Motion And Control Ring Optical) és a Firewire alapú hajtás. Ezek közül a legelterjedtebb a SERCOS az egyetlen, mely mind fizikai, mind kapcsolati szinten szabványos megoldást jelent, miközben a hajtások mellett a beavatkozók és szenzorok is rákapcsolhatók. Létezik egyfajta átfedés a másik oldalon is, hiszen a terepi buszok között is sokféle nyílt megoldás ismert (Profibus, FIP, CAN stb.), amelyek ugyanakkor alkalmasak hajtások rendszerbe illesztésére is (pl. Profibus).

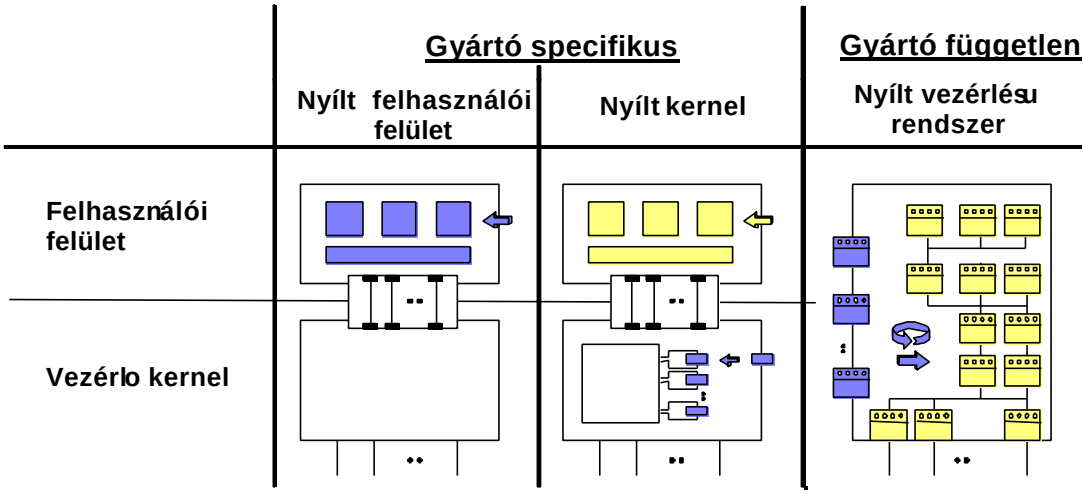


5. ábra Vezérlés lehetséges külso kommunikációs kapcsolatai

Egyértelmu igénye a felhasználóknak az NC programok G kódjainak a leváltása egy korszerubb programozási nyelvre (a 4. táblázat 5. pontja). Ezen kérdés kezelése azonban némileg elkülönül a nyílt vezérlési architektúrák kérdésétol. Az OMAC projektben felismerték, hogy a két kérdés összetartozik, és kialakítottak egy ezzel foglalkozó önálló munkacsoportot. A japán OSEC projektben pedig OSEL (Open System Environment Language) néven a G kódokat meghaladó megoldást [134] fejlesztettek ki. Az ezzel kapcsolatos további kutatásokat az 5.2 alfejezet mutatja be.

4.1 A NYÍLT ARCHITEKTÚRÁJÚ VEZÉRLÉSEK ÁTTEKINTÉSE

A 6. ábra a nyílt vezérlok osztályba sorolását mutatja aszerint, hogy belso felépítésük mennyire nyílt:



6. ábra A nyílt vezérlések belso felépítésének három lehetséges felépítése [1]

1. A nyílt felhasználói felület jelenti a legkisebb mértéku nyíltságot a vezérlok piacán. Ez a lehetoség ma már az eladhatóság miatt minimális követelmény és az összes nagy gyártó (pl. GE Fanuc, Siemens) valamilyen formában biztosítja. Tipikus megoldás, hogy a felhasználói felület egy önálló, valamilyen MS Windows-t futtató PC, amelyen hagyományos Windows programokat is lehet futtatni, és megfelelo feltételek mellett módosítható a felhasználói felület, a kernel egyes változói lekérdezhetok alkalmazói programokból.

   A GE Fanuc, a világ legnagyobb CNC gyártója szerint például a nyílt CNC egyet jelent a PC-be integrált CNC-vel [63]. Többféle megoldást kínál, ahol a CNC maga egy PC-be dugható kártya, magán a PC-n Microsoft NT vagy CE operációs rendszerrel. A felhasználó vagy a szerszámgépgyártó saját programjait is futtathatja a rendszeren, melyek elérhetik a CNC-t egy jól definiált programozói interfészen keresztül.

2. A nyílt felhasználói felület mellett lehetoség van a vezérlo kernel funkcióinak kiegészítésére, egyes algoritmusainak lecserélésére. Ezt a nagy vezérlésgyártók csak nagyon korlátozott mértékben támogatják.

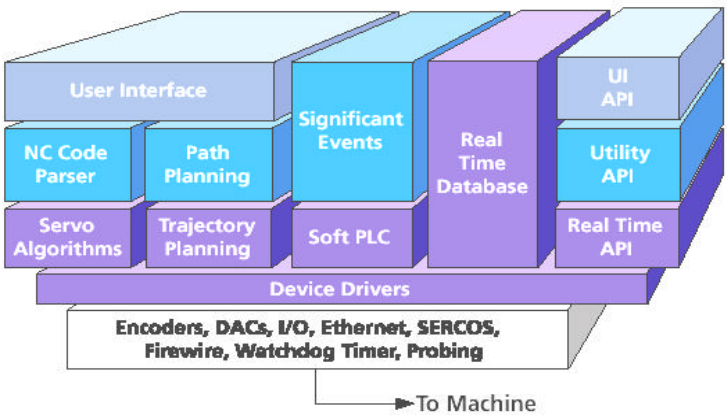
   Európában a másik piacvezeto a Siemens egyik alap CNC-je, a SINUMERIK 840C a következo funkcionális elemekbol épül fel: MMC (ember-gép kapcsolat), NCK (NC kernel), PLC. A nyíltság MMC szinten az MS WinNT alapon támogatott OEM szoftverek futását jelenti, vagyis ugyanaz a megoldás, mint a GE Fanuc esetében. Korlátozottan - OEM szerzodéssel - a SINUMERIK nyílt architektúra lehetové teszi, hogy speciális - OEM - NCK funkciókat építsenek a kernelbe, mely érintheti az interpolátort, a pozíció kontrollt stb. Mind a CNC, mind a hozzátartozó PLC többféle lokális hálózati megoldást támogat (OPC, TCP/IP, S7, Profibus, MMS) különbözo mértékben. [155]

A "nagyok" mellett egyre több, kisebb cég ajánl komplett PC alapú, korlátozottan nyílt CNC megoldást. Az architektúra szempontjából háromféle megoldás ismert:

a) A PC alaplapja mellett egy kiegészíto kártyán fut a vezérlo real-time része (pl. DSP jelprocesszoron) közvetlen szervo és I/O csatlakozással [35, 116]. Sok más termék közül az egyik legismertebb a Motion Engineering XMP [123] családja, amely rendszerben a real-time részek egy C/C++ interfészen keresztül elérhetok, hogy a PC processzorán bármilyen alkalmazói programot lehessen a rendszerhez hozzáépíteni. Hasonló elven muködik a magyar NC Technika új fejlesztésu eszterga és maróvezérlése [126], amelynek azonban jelenleg nincs nyilvános API felülete.

b) Két vagy több processzoros konfiguráció, amikor az egyiken egy real-time rendszeren fut a gépvezérlo kernel, a másikon pedig Windows operációs rendszer alatt a felhasználói felület, hálózati kapcsolat stb. (pl. [53]). Ez a modell követi a hagyományos struktúrát, melyet a Siemens és a Fanuc is követ, de van ilyen KUKA robot is. A Hewlett-Packard, amely általában elkötelezett a nyílt rendszerek irányában, MS WinNT és LynxOS alapú nyílt vezérléseket forgalmaz Sercos hajtásokkal és nyílt C++ API-val.

c) Egyetlen alaplapon, az eroforrásokat megosztva fut a real-time és a Windows operációs rendszer. Ilyen elven muködo robotvezérloket (KR C1) árul sikeresen többek között a német KUKA cég [149]. Az MDSI OpenCNC® [108] rendszerében MS WinNT és RTX real-time operációs rendszer fut ugyanazon a processzoron. Az 7. ábra jobb oldalán látható, hogy a rendszernek van NT és real-time alkalmazói interfésze is. A mai PC-k mellett ez az architektúra muködoképes, hiszen 450 MHz-es Pentium III-on a szervo frissítési ido 10 tengely esetében maximum 667 µs-re adódik. Ismert hagyományos DOS alapú megoldás is [148].



7. ábra Az OpenCNC® moduláris felépítése

3. Végül a teljesen nyílt vezérloket azok a rendszerek jelentik, amelyeket valamilyen nagyobb kutatási konzorcium fejlesztett ki gyártótól függetlenül, illetve több gyártó bevonásával. Ezeket mutatja be részletesen a következo alfejezet.

4.2 A LEGFONTOSABB NYÍLT VEZÉRLO INICIATÍVÁK

Ezek a nyílt vezérlo struktúrák ténylegesen gyártó független architektúrák, és így a többé - kevésbé nyílt specifikáció nyomán bárki készíthet ilyen típusú vezérloket. A fenti nyílt vezérlok a vezérlogyártók, a szerszámgépgyártók és a végfelhasználók számára is sok elonyt jelentenek [143, 45]. Az alábbiak a legfontosabb tulajdonságaik:

- moduláris topológia, a modul-osztályok szabványos interfésszel rendelkeznek,
- a modulok lecserélhetok hasonló - akár más gyártótól származó - modulra,
- új szabványos modulok építhetok hozzá, a végfelhasználó számára az ún. "testre szabás" könnyen megoldható,
- topológia és teljesítmény szerinti skálázhatóság,
- egyik környezetbol a másikra való portolhatóság (áttelepíthetoség), újrahasznosítható programkódok,
- speciális algoritmusok egyszeru integrációja.

Kisebb hangsúlyt kap, de mindenképpen fontos látni a nyílt vezérlések szempontjából kritikus tényezoket, amelyek gátolják az ilyen rendszerek gyors elterjedését:

- a nagy vezérlésgyártók egyfajta ellenérdekeltsége,
- ki vállal felelosséget az egész rendszerért,
- kérdéses az ilyen vezérlok megbízhatósága,
- eleinte az ár biztosan magasabb, mint a hagyományos vezérléseké.

A fentiek közül a felelosségvállalás kérdését tartom a legkritikusabbnak, hiszen a különbözo gyártótól beszerzett nyílt vezérlo esetében az is kérdés, hogyan állapítható meg az esetleges hibás muködés nyomán a gyártó felelossége. Ez másképp azt is jelenti szerintem, hogy ipari alkalmazásokban nem várható az elemeiben teljesen más - más gyártó termékeibol "összerakott" CNC megjelenése.

A legjelentosebb az európai OSACA [143,1], az észak-amerikai OMAC [45, 130] és a japán OSEC [60, 133] projekt. Természetesen ezek mellett is léteznek kevésbé ismertek:

- Az OSAViS (Virtuális Muhely Nyílt Rendszeru Architektúrája), amely a virtuális gyártás egyik legismertebb kísérleti szoftverének, a japán VirtualWorks-nek továbbfejlesztése, szintén alkalmas valódi vezérlések nyílt összekapcsolására [88].

- Sok szállal az OMAC-hoz kötodo, de önálló kutatás a Koren vezette U-M OAC (University of Michigan Open Architecture Controller) [96]. Az évek során sokféle szervovezérlést, kompenzációs algoritmust és interpolátort fejlesztettek hozzá. A logikai felépítése alapján muszakilag az egyik legjobb nyílt architektúra, amelynek 11 függvényosztályba tartozó moduljai között diagnosztikai és modellezési feladatot ellátóak is találhatóak.

- A NIST-ben fejlesztették ki az EMC (Enhanced Machine Controller) rendszert, ami forráskódban (C/C++) szabadon letöltheto az Internetrol [153], ezért sok

kísérleti környezet alapját képezi. Modul felépítése nagyon hasonlít az OMAC API-ra, mert annak egyik alapja.

Ugyancsak léteznek olyan közös kutatási-fejlesztési konzorciumok, amelyek témája eros átfedésben van a nyílt vezérlésekkel, de fo célkituzésük más:

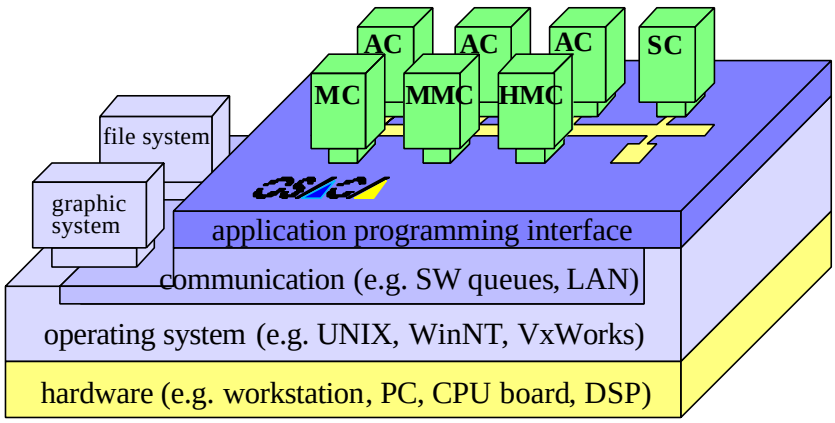
- A NIIIP (National Industrial Information Infrastructure Protocols) [128] vállalati modell egy magasabb szintjén tuzött ki hasonló célokat, hiszen virtuális vállalatok számára eroforrás, interfész és protokoll referenciákat alkot, amelyekkel ezek kooperálása hatékonyabb lehet. Felhasznált eszközkészletében (Corba, STEP) sok rokon vonást mutat a nyílt gépvezérlési iniciatívákkal.

- Sematech konzorcium félvezetogyártó cégek számára dolgozta ki az ún. CIM Keretrendszerét [151], amelynek tapasztalatai azonban gépipari alkalmazásokban is hasznosíthatók.

4.2.1 OSACA

Az OSACA (Open System Architecture for Controls within Automation Systems) projekt 1992-ben indult ESPRIT támogatással. 1998-ig 96 emberév és 12,3 millió Euro költségvetéssel kifejlesztett egy gyártó független nyílt vezérlést a világon elsoként. Az OSACA konzorciumban Európa jelentos vezérlésgyártói (Siemens, Bosch, NUM, Fagor, stb.), több szerszámgépgyártó (Homag, Comau, Trumpf stb.) és kutatóintézet (ISW, WZL, IWB, INTEC stb.) mellett a BMW és a DaimlerChrysler, mint végfelhasználó kapcsolódott be a munkába. Magyarországról az OSACA harmadik fázisának indulása óta, 1997 elejétol a SzTAKI vesz részt a munkában.

Az OSACA rendszer architektúra két alapveto elembol áll. Az egyik egy olyan általános alkalmazói program interfész (API), amely a hardvertol, az operációs rendszertol és a modulok közötti kommunikáció módjától független csatlakozási felületet kínál az OSACA moduloknak (AO, application object) (8. ábra). A másik, hogy az ún. OSACA kézikönyv [1] részletesen specifikálja az OSACA referencia modellt, vagyis azokat az OSACA modulokat, amelyekkel egy CNC felépítheto.



8. ábra Az OSACA platformra csatlakozó CNC modulok (AO-k)

Az OSACA API három rétegbol áll: az üzenettovábbító réteg (MTS), az alkalmazás szervizek rétege (ASS) és a kommunikációs objektumok kezeloje (COM). Az MTS biztosítja az alapveto szolgáltatásokat (üzenetek küldését és fogadását), bármilyen hardver - szoftver - kommunikációs platformon realizálható, és a kapcsolatfelvételhez, kapcsolattartáshoz és kapcsolatfelbontáshoz tartozó feladatokat is ellátja. Az ASS állítja össze és szedi szét az alkalmazói üzeneteket, kódolja és dekódolja azokat, ha szükséges. Az alkalmazásfejleszto a COM segítségével éri el az OSACA API-t. Minden egyes AO-nak megvannak a saját objektumai, amelyek a COM-on keresztül elérhetoek. Az objektumok háromfélék lehetnek: változók, események és metódusok.

Az OSACA referenciamodell legfontosabb része az NC kernel funkcionális egység, amelynek moduljait részletesen specifikálja. Ezek a következok: tengelyvezérlok (AC), mozgásvezérlo (MC), fotengelyvezérlo (SC), mozgásvezérlo szervezo (MMC). Az OSACA definiál ugyan, de részletesen nem specifikál még egy ún. logikai funkcionális egységet, amely a PLC-szeru funkciókat tartalmazza.

OSACA vezérléseket több egyetemi környezetben, kiállításokon (pl. EMO Hannover) valamint prototípusokat a BMW és a Mercedes gyárában is bemutattak az évek során. A Siemens bejelentette, hogy piacra fog dobni OSACA vezérlést, de ennek részletei még nem ismertek.

A SzTAKI-ban Drozdik fejlesztett ki az OSACA alapjain álló, de újszeru szoftvermegoldásokat alkalmazó kísérleti vezérlést (DAC, Distributed Architecture Controller) [52]. A DAC-ban a nem real-time részek CORBA technológiával kapcsolódnak egy hálózatra, és egy ún. OSACA-CORBA gateway-en keresztül jutnak el az adatok a real-time részekhez. A konkrét implementáció Java és C++ modulokból áll és egy 3 tengelyes marógépet vezérel.

Az OSACA eredményeit értékelve rá kell mutatni néhány általános problémára is, amelyek miatt nem fejlodik megfeleloen:

- Az OSACA magán viseli az EU finanszírozás korlátait, így 1998 óta nincs pénz a továbblépésre, lassan elhalnak az eredmények, jelentos felhasználónak egyes egyetemi környezetek számítanak, ahol minden CNC kutatás OSACA platformot használva folyik (WZL Aachen, ISW Stuttgart). Az ún. OSACA Association nem jelent elég financiális hátteret a vitathatatlanul elso nyílt iniciatívának, hogy továbbléphessen. Igazolásul elég meglátogatni a www.osaca.org honlapot.

- Szemben az OSEC - JOP és az OMAC munkákkal (4.2.2 és 4.2.3 alfejezet) az OSACA-ban nem kapcsolódik össze a nyílt vezérlés belso architektúrájának kifejlesztése a többi nyíltsággal kapcsolatos feladattal (4. táblázat).

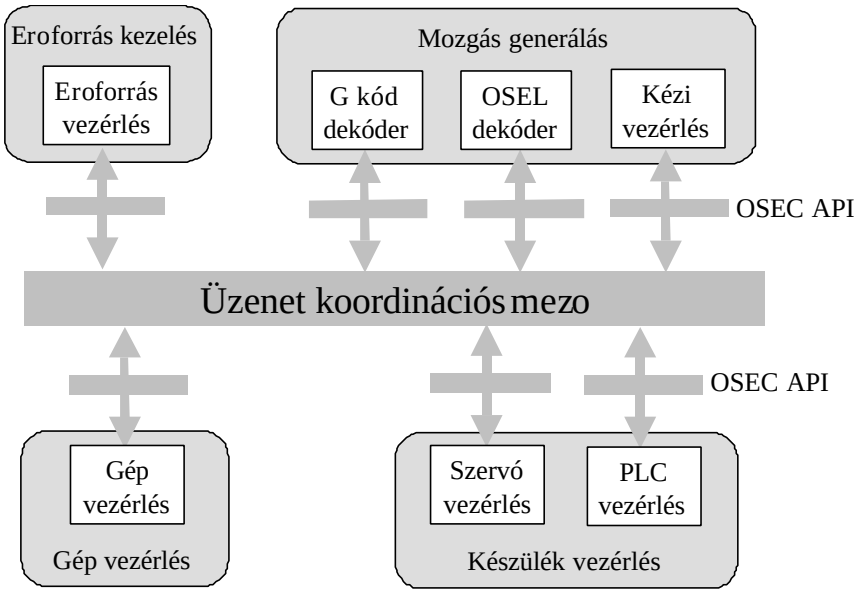
4.2.2 OSEC - JOP

1994 decemberében a Toyoda, a Toshiba, a Mazak és a Mitsubishi szoftver cégekkel közösen alakította meg a japán nyílt vezérlés kifejlesztéséért felelos munkacsoportot OSEC (Open System Environment for Controllers) néven [133]. Később sok más japán vállalat és egyetem csatlakozott, amelyek korábban az ún. nyílt FA (Factory Automation) vezérlohálózat kifejlesztésén dolgoztak együtt és így megalakult a JOP

(Japan FA Open Systems Promotion Group) [89], ahol több munkacsoportban próbálnak a nyílt gyártás különbözo kihívásaira válaszokat kidolgozni.

Megalkották az ún. OSEC referencia modellt, ami az OSI hálózati rétegek analógiáját követve 7 rétegbol áll. A rétegek a következok: mechanikai, elektromos, eszközvezérlo, geometria-vezérlo, kommunikációs, muveleti és tervezo, valamint CAD/CAM.

Az OSEC modellben a rétegek közötti adatáramlást vették alapul, így a figyelem középpontjában az egyes rétegek között kapcsolatot teremto programozói felület áll. Az 1998-ban elkészült OSEC API [134] a vezérlés funkcionális moduljai között egy ún. üzenet koordinációs mezot feltételez, ezen keresztül tudnak a modulok kapcsolódni (9. ábra).



9. ábra OSEC referencia architektúra

Az OSEC API definiál néhány általános objektumot, valamint az egyes modulokhoz eljárásokat, amelyeken keresztül érik el más modulok az adott modul szolgáltatásait. Az API egyszerusítése érdekében opcionálisak azok az eljárások, amelyek szofisztikusabb, bonyolultabb jelentéssel bírnak, de nem feltétlenül szükségesek. Így egy minimum API készlettel is építheto muködoképes OSEC vezérlo.

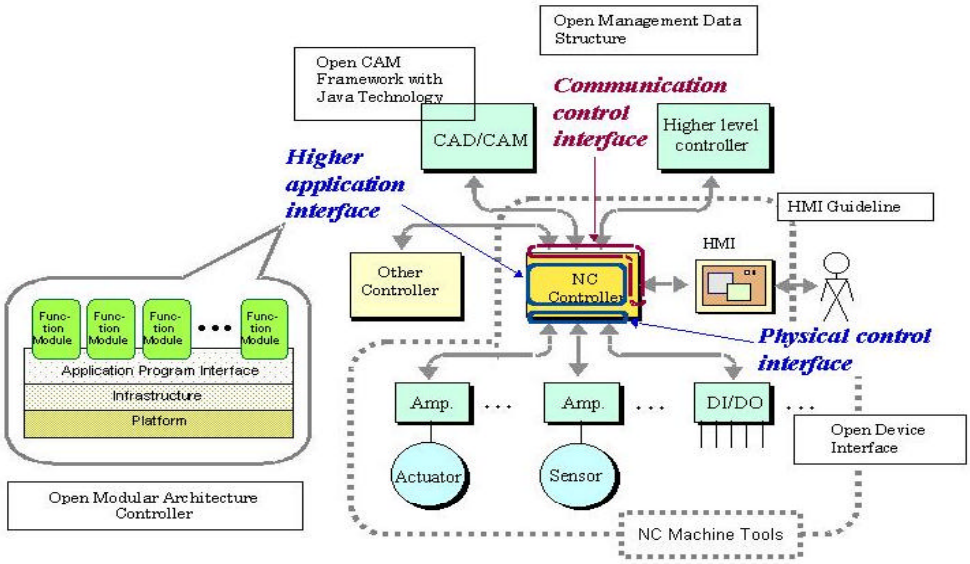
Ugyancsak a JOP keretében dolgozták ki a 1999-ben az ún. PAPI specifikációt [90], amely a CNC alkalmazói program interfésze, vagyis a vezérlés és a megjeleníto (HMI) közötti kapcsolatot specifikálja.

A JOP mára egyre több feladatot fog össze, és az elkészült analízisük összefoglalja a nyílt vezérlések problémáit (10. ábra), amelyeket önálló csoportokban dolgoznak fel.

Kritikusan értékelve a JOP tevékenységét, az alábbi megállapításokat kell tenni:

- A PAPI és az OSEC API egymással semmilyen módon nincsenek összehangolva, átfedések ellenére a funkciók között nem lehet egyszeru

megfeleltetéseket találni. Inkább két egymástól független munkacsoport eredményének látszik, noha a ketto publikálása között mindössze egy év telt el. Ez a párhuzamosság a JOP-on belüli munka koordinálását kérdojelezi meg, az esetleges ellenérdekeltségek jelenlétét is tükrözheti.



10. ábra Nyílt vezérlo megoldandó problémái (JOP analízise) és csoportosítása

- A JOP keretében kifejlesztett FL-net hálózat [91] UDP/IP protokoll felett javasol megoldást nyílt ipari hálózat számára vezérlések összekapcsolására. Az UDP szint tetején egy ún. FA Link biztosít tokenes megoldást ciklikus és üzenet jellegu adatátvitelre. A javaslat elorevetíti a MAP történetének a megismétlodését, hiszen ugyanaz a problémája, ami a MAP sikertelenségét is okozta. Az ipar nem fogja átvenni az olyan hálózati technológiát, amely nem feltétlenül szükséges, ha a szélesebb körben elterjedt meglévo megoldással ugyanazt el lehet érni. Semmi se indokolja a hagyományos információ hálózat és a terepi (ill. digitális hajtás) hálózat közé egy újabb hálózati szint integrálását.

- Nagyon változó - legalábbis ezt tükrözi az Internetes honlapjuk -, hogy az egyes munkacsoportok milyen mennyiségu és minoségu munkát végeznek, és azt mennyire publikálják.

4.2.3 OMAC

Az Észak-amerikai OMAC (Open Modular Architecture Controller) kezdete 1994-re nyúlik vissza, amikor a három nagy amerikai autógyár publikálta a jövobeli vezérlésekkel szemben támasztott közös követelményrendszerét [45].

Az Egyesült Államok által szponzorált TEAM (Technologies Enabling Agile Manufacturing) program keretében kifejlesztett ún. TEAM vagy OMAC API-ban [130] egyedülálló, hogy nem definiáltak egy rögzített referencia architektúrát, hanem helyette egy általános keretrendszert és módszertant dolgoztak ki, amelyben a következo fogalmakat definiálták:

- Az ún. alaposztályok (Foundation Class) egy általános vezérlo architektúra osztályokra bontásából (dekompozíciójából) keletkeztek.

- Az alaposztályok modulokba (5. táblázat) vannak csoportosítva, amelyek az OMAC vezérlo építoelemei, komponensei. Egy adott vezérlés felépítése ezek után a különbözo OMAC modulok egyedi implementációiból áll. A modulok azok, amik rendelkeznek jól definiált programozói felülettel (API), belso állapotokkal és állapot átmenetekkel. A fejlesztonek szabad keze van a rendszer architektúra kialakításában, ebben az OMAC csupán útmutatásokkal szolgál.

- A modulokat IDL (Interface Definition Language) nyelven specifikálták, amely nyitva hagyja az implementálás kérdését. Az OMAC komplex infrastruktúrája - saját értelmezése szerint - jelenti a hardver platformot, az operációs rendszert, fordító + szerkesztési eszközöket és az elosztott futtatási környezetet (pl. DCOM, Corba stb.). A konkrét implementációk számára szintén teljes szabadságot ad az OMAC ezekben a kérdésekben. (*Megjegyzem, hogy ez a legutóbbi idoben hátrányosan változott, lásd az MIDL-lel kapcsolatos problémát alább*.)

- Az egyes modulok belso felépítésének a realizálását is elore specifikált építoelemekkel támogatják, azoknak a moduloknak, ahol ezt meg lehet tenni. A modulok belso állapotát egy-egy véges állapotgéppel (FSM) írják le (pl. Axis, Task Coordinator).

Axis	**Control Law**	**Human-Machine Interf.**	**Process Modell**
• egyetlen tengely mozgásvezérlése • használja Control Law-t • szervo kiegyenlítés • állapotok	• trajektória követés • erosítés szabályozás	• indítás/leállítás • üzemmód választás • konfigurálás • diagnosztika • karbantartás • setup	• elotolás felülírás • fotengely sebesség felülírás

Axis Group	**Control Plan**	**IO Points**	**Task Coordination**
• több tengely összehangolása • look ahead • sebesség profil generálás • feedhold; stop	• FSM specifikálás • vezérlési utasítások szekvenciája • konkrét vezérlési parancsok	• adat írás/olvasás • adat ciklikus küldés • adat jóváhagyás • szenzor integráció	• FSM specifikálás • indítás/leállítás szekvenciák • taszk koordinálás • vezérlési ciklus • hiba naplózás

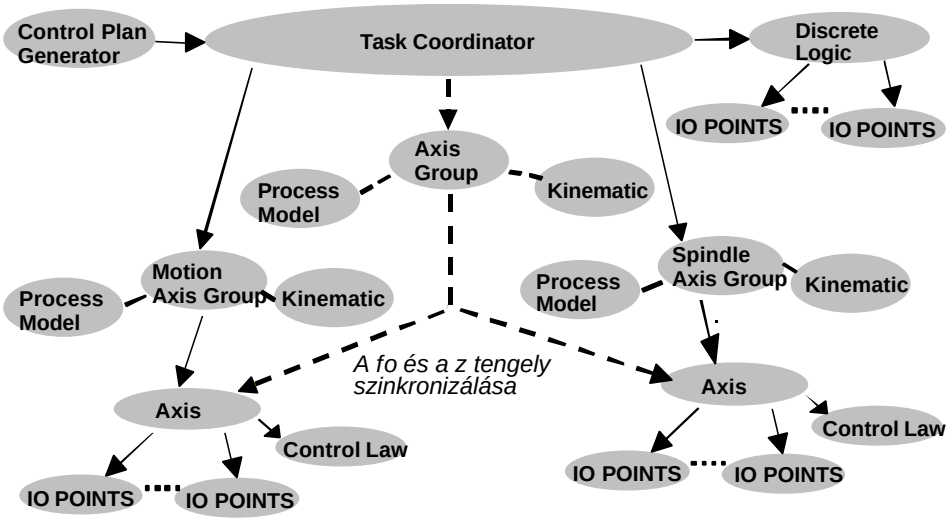
OMAC Base	**Control Plan Generator**	**Kinematics**
• név, verzió kezelés • név és mappa szervizek	• IEC1131, RS274D lefordítása • vezérlési utasítások generálása	• kinematikai számítások • koordináta transzformáció • szerszám korrekció • kompenzálások

Capability	**Discrete Logic**	**Machine-to-Machine**
• vezérlési utasítások koordinációja • NC muködési módok •	• FSM specifikálás • 1131 jellegu parancsok végrehajtása • mód kapcsolás	• távoli elérés • fájl transzfer hálózaton • program invokáció • esemény monitorozás

5. táblázat OMAC modulok [130]

Az OMAC API teszteléséhez a NIST, a Michigan-i egyetem és a LLNL (Lawrence Livermore National Laboratory) építettek fel nyílt gépvezérlést, amelyek ugyanabból az interfész specifikációból (IDL) más-más szoftvertechnológiai megoldásokkal építkeztek. Az LLNL rendszerének tesztelésébe a SzTAKI is bekapcsolódott 1997-ben [12]. Az LLNL rendszere az IDL leírás alapján Java nyelven írt modulokból áll. A vezérlés real-time részét a Java-ból C-re automatikusan lehet átfordítani speciális eszközökkel. A rendszer egy a Windows mellett/alatt futó real-time kernelen és a Windows-on futó Sun JVM-en (Java Virtual Machine) muködik.

Az OMAC modulokból (5. táblázat) az egytengelyu rakodószerkezettol vagy PLC-tol kezdve komplex soktengelyes megmunkáló központig vagy sorig bezárólag mindenféle vezérlés felépítheto. A következo ábrán egy fúróberendezés vezérlése látható, ahol a Z tengely mozgása és a fotengely forgása szinkronizálható.



11. ábra Fúrógép OMAC moduljai [130]

Az API definiálása mellett jelenleg az OMAC keretében további munkacsoportok is muködnek, amelyek feladatai a következok: (1) globális HMI specifikálása; (2) a Microsoft környezetben megvalósított real-time környezet vizsgálata; (3) újfajta NC program nyelv és az OMAC összehangolása; (4) PC alapú vezérlések stb.

Az OMAC-kal kapcsolatban az alábbi hiányosságok mondhatók el:

- Az API már a 0.23 verziónál tart hosszú évek munkája nyomán, de még mindig nem készült el az 1.0 változat. Másfelol a jelenlegi API elképesztoen bonyolult (pl. a tengely (Iaxis) modulhoz több mint 10 csoportba foglalva közel 400 eljárás tartozik).

- Az 1994-es követelményrendszer óta 7 év telt el, és még csak kizárólag laboratóriumi demonstrációk ismertek OMAC alapú CNC-kkel.

- Nem létezik pontos határidokkel kiegészített menetterv, a beígért munkaanyagok terén is rendszeres csúszások tapasztalhatók.

- Az utolsó OMAC API verziók IDL helyett MIDL-t (Microsoft IDL) használnak, ami nem szabványos és így nehezíti az OMAC modulok nem Microsoft platformra való implementálhatóságát.

4.2.4 A NYÍLT VEZÉRLÉSI JAVASLATOK ÖSSZEHASONLÍTÁSA

Az eddigiekben bemutatott legfontosabb három nyílt vezérlés mindegyike általános megoldást akar adni a CNC technológiában a nyíltság kérdésére. Az OMAC keretén belül létezik arra erofeszítés, hogy a három projekt eredményeit összehangolják, esetleg egy IMS javaslat elkészüljön. Annak az eldöntéséhez, pontosabban megbecsüléséhez, hogy mi várható a nyílt vezérlésekkel a közeljövoben, nem elhagyható a három iniciatíva eredményeinek az összehasonlítása és a tapasztalatok kiértékelése.

Részletes összehasonlítás nem ismert, jelenleg az OMAC keretében a HMI specifikációk összevetése zajlik. Az OMAC és OSACA egyfajta összehasonlítását Lutz végezte el [107], o azonban megmaradt az általánosságok szintjén (mindketto objektum orientált, kliens-szerver architektúrára épít stb.), cikkének végso kicsengése: "nem is különböznek annyira".

Az alapproblémát egyértelmuen az jelenti, hogy a fenti - bizonyos szempontból még kiforratlan - nyílt rendszerek egymással sem kompatibilisek, vagyis konkurenciát jelentenek egymásnak. Kicsit hasonló a helyzet, mint egy-egy dominens gyártónál, vagyis bármely nyílt rendszer azt "várja", hogy a többi megoldás alkalmazkodjon és vegye át specifikációját. Sajnos közös probléma mindhárom esetben, hogy a referencia architektúra és az API kifejlesztése túlságosan kis csapat munkája, magán viseli a projektbe bevont gyártók korábbi termékeinek a logikáját (pl. az OSACA a Siemens vezérlését).

Az összehasonlításhoz [18, 33] azokat a szempontokat kell összevetni, amelyeknek mindegyikben feltétlenül szerepelniük kell. A részletes vizsgálatok alapján ezeket az alábbiakban tudom összefoglalni:

- az alkalmazói program interfész logikája, mérete, részletezettsége,
- a referencia architektúrában definiált modulok kapcsolata és azok belso muködése,
- a mindenképpen szükséges funkciók modulokba sorolása,
- az infrastruktúra (hardver, szoftver, hálózat stb.), amely szükséges a vezérlo felépítéséhez,
- az egyéb nyíltsági jellemzok (4. táblázat).

4.2.4.1 Alkalmazói program interfész (API)

Elsoként, az API-kat vizsgálva formális és némileg szubjektív (elony, hátrány) megállapítások tehetok. A vizsgálatban korlátot jelentett, hogy az OSEC esetében mindössze a hívások neve állt rendelkezésre és az elnevezésekbol kellett a függvények valódi tartalmára következtetni.

Kritikus, hogy a három API logikája eltér egymástól, a legmodernebb az OMAC, a másik ketto más-más okok miatt szoftvertechnológiailag elavultnak tekintheto.

API	OSACA	OSEC - JOP	OMAC
Logikája	objektum orientált	függvény hívások	komponens alapú
Definíciós nyelve	C++	C	IDL
Részletezettség	kicsi	nagyon kicsi	nagy
Mérete	~ 100 objektum, esemény és eljárás	~ 150 hívás	több ezer hívás, több száz típusdeklaráció
Elonyei	granularitás	könnyen értheto	teljes köru
Hátrányai	eljárások helyett változók, union jellegu definíciók, bonyolult elnevezések	nem objektum orientált, hiányos	rendkívül bonyolult, nincsen minimum készlet

6. táblázat Nyílt vezérlések programozói felületének (API) összehasonlítása

Lényegi különbség a granularitásban figyelheto meg, az OMAC sokkal részletesebben, elemekre szedve definiálja az interfészét. A másik ketto, különösen az OSEC elnagyolt. Pl. a tengely modul esetében az OSACA és az OSEC nem tartalmaz sok olyan elemet, ami fontos. Az OMAC viszont annyira bonyolult, hogy az már az alkalmazhatóságát kérdojelezi meg.

4.2.4.2 Referencia architektúra

Bár a dokumentációk szerint az OMAC vezérlésnek nincs referencia architektúrája, ez a valóságban azt a szabadságot adja, hogy a definiált modulokból a fejleszto szabadon állíthatja össze a vezérlését. Ezt a szabadságot viszont a másik két architektúra is biztosítja, így a három architektúra ténylegesen összehasonlítható.

	OSACA	OSEC - JOP	OMAC
Referencia architektúra	létezik	létezik	formálisan nincs
Modulok száma	9	7	14
Részletesen specifikált modulok száma	4	7	14
Alkalmazhatóság	közepes	közepes	széles
Modulok muködési leírása	belso muködés definiálatlan	belso muködés definiálatlan	belso muködés leírása (véges állapotgép FSM)

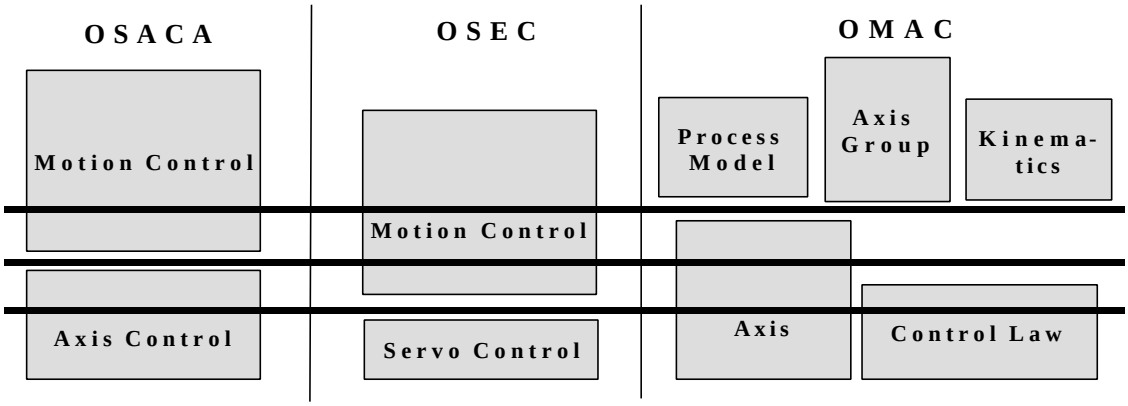
7. táblázat Nyílt vezérlések referencia architektúrájának összehasonlítása

Az összehasonlításom arra is választ keres, hogy mennyire kompatibilisek a referencia architektúrák egymással, van-e reális lehetoség arra, hogy a három iniciatívát jelen formájukban egységesíteni lehessen. Ennek érdekében a vezérlés egyik kulcs alkotórészét, a tengelyvezérlést a három megoldásban összehasonlítottam. Ehhez mind a három nyílt vezérlés közzétett alkalmazói program interfészét (lásd. a Függelék 11.1, 11.2, 11.3 és 11.4 részében) kellett elemeznem. Az API-k hívásait összevetve, ahol ez lehetséges az egymásnak megfelelo hívásokat párosítva, a vezérlok dekompozíciójával keletkezett modulok határai válnak összehasonlíthatóvá. Az OMAC esetében az Axis modulhoz szorosan kapcsolódó Control Law modult is bele vontam a vizsgálatba, mivel olyan feladatokat lát el, amelyek a másik két esetben beletartoznak a tengely modulba.

A 12. ábra szemlélteti az összehasonlítás eredményét, hogy a három nyílt vezérlo esetén a dekompozíció más-más logikai szinten húzta meg a modulok határait, aminek következménye, hogy a referencia architektúrák teljes mértékben inkompatibilisek. Például az OSEC-ben a mozgásvezérlo (Motion Control) modulhoz tartoznak a

tengelyvezérlés hívásai, míg az OSACA és OMAC esetében a szervo és a tengelyhívások vannak egy modulban. Hasonlóan az OMAC jog vagy home hívásai nem tengelyfunkciók az OSACA vagy az OSEC esetében. Az OSACA egyedüliként megkülönbözteti a fotengelyt a többitol (Spindle Control). Az OMAC az egyetlen, ahol interfész szinten elérheto a sebesség és áram kontroll, a másik kettonél kizárólag a pozíció.

Reménytelen tehát a meglévo megoldások olyan finom módosítása, hogy egy jövobeni általános nyílt vezérlo architektúra alakulhasson ki e három megoldás összeolvasztásával. Ez utóbbi cél értelmesen csak valamelyik megoldás egyfajta kitüntetett szerepe mellett képzelheto el a másik ketto rovására.



12. ábra A három nyílt vezérlés tengely moduljainak logikai összehasonlítása

4.2.4.3 Infrastruktúra

Az infrastruktúra itt a teljes hardver-szoftver környezetet jelenti, ahol az adott vezérlés fejleszthetо, futtatható. Mivel mindegyik rendszer nyílt, ezért kis eltérések találhatók közöttük, de van egy lényeges, alapveto különbség. Egyedül az OMAC hagyja teljes egészében nyitva az infrastruktúra kérdését, míg a másik ketto célkituzéseivel ellentétesen egyedi szoftvermegoldással maga generált ún. futtatási platformot. Ez a lényegi különbség azt jelenti, hogy az OSACA és OSEC szabad felhasználása mindig a platformot realizáló szoftver kérdése lesz, vagyis kötodik szervezethez, eroforráshoz, érint tulajdon és licence jogokat. Az OMAC-nál viszont egyszeruen elegendo az IDL fájlokban leírt modulok és hívások betartása egy szabadon választott környezetben, ami lehetové teszi, hogy az aktuálisan korszeru infrastruktúrát használva futhasson az OMAC vezérlo.

	OSACA	**OSEC - JOP**	**OMAC**
Kommunikációs platform	OSACA specifikus	speciális dll-ek	bármi lehet
PC alapú	nem csak	igen	nem csak
Operációs rendszer	VxWorks, Windows, OS9, Solaris	Windows, ???	bármi
Program nyelv	C/C++	C/C++	C/C++, Java, …
Elosztottság	elosztott	nem elosztott	elosztott
DCOM, Corba, stb.	nem	nem	igen
Informatikai korszeruség	átlagos	kissé elavult	eloremutató

8. táblázat Nyílt vezérlések hardver/szoftver környezetének összehasonlítása

4.2.4.4 Összegzo értékelés

Bemutatók szempontjából az OSACA kiemelkedik, több kiállításon és ipari környezetben muködött prototípusa. A nyíltság egyéb kérdései (4. táblázat: felso-, alsószintu kommunikáció, NC program nyelv) tekintetében vegyes a kép, egyik iniciatívát sem lehet különösebben kiemelni.

	OSACA	**OSEC - JOP**	**OMAC**
Bemutatók	több nyilvános (1996-98)	laboratóriumi (1999)	laboratóriumi (2000)
Felsoszintu kommunikáció	említi az MMS-t [2]	nincs	említi az MMS-t
Alsószintu kommunikáció	Sercos [3]	???	Sercos
Nyílt NC programnyelv	nincs	OSEL	STEP-NC

9. táblázat Nyílt vezérlések egyéb tulajdonságainak összehasonlítása

Így az összehasonlításban egyértelmuen bebizonyítottam, hogy a három iniciatíva gyors és egyszeru összeolvadása nem lehetséges, a definíciókban (API, referencia architektúra infrastruktúrában) rejlo ellentmondások miatt.

Ugyanakkor semmiképpen sem jelenti a fenti gondolatmenet és összehasonlítás a nyílt vezérlések elvetését, hiszen a kezelni kívánt problémák megvannak és egyre inkább zavarók a gyártásautomatizálásban. A XXI. század gyártórendszer architektúrájának jósolt Újrakonfigurálható Gyártórendszer (RMS) [97] egyik alapeleme is csak egy nyílt, átkonfigurálható vezérlés lehet, ami valamilyen módon ezekbol a mostani kutatásokból fog kinoni.

4.2.5 EGYSZERUSÍTETT OMAC API

Az inkompatibilitás megoldása után azt vizsgáltam, hogy melyik iniciatíva a leginkább eloremutató. Az összehasonlítás már megmutatta, hogy az informatikai korszeruség és a részletes modellalkotás szempontjából egyértelmuen az OMAC mellett érdemes letenni a voksot. Ugyanakkor az OMAC-ra kevés implementáció a jellemzo, még prototípus szintu sincs sok. Véleményem szerint ennek oka az OMAC jelenlegi API-jának elképeszto bonyolultságában rejlik, amely megkerülhetetlen a szélesköru alkalmazhatóság szempontjából. Nem lehet arra számítani, hogy a több ezer hívásból álló API specifikáció mellett kiálljanak a gyártók.

	OSACA	**OSEC - JOP**	**OMAC**
Fo hátrány	1998 óta nem fejlodik	OSEC ← →PAPI	egyre bonyolultabb
Fo elony	sok kísérleti alkalmazás	egyszeruség	korszeru szoftverháttér

10. táblázat A különbözo nyílt vezérlések fo hátránya és elonye

Megoldásnak javasolom [18, 33] az OMAC interfészek és eljárások felosztását minimum készlet és bovítési (vagy opcionális) kategóriákba, ami az egyes OMAC interfészeknek csak egyes elemeit tartja meg kötelezonek, a többit opcionálisnak kezeli. A minimumkészlet kiválasztásánál arra kellett törekedni, hogy

1. a modulok alapfunkciói megvalósíthatók legyenek a minimum készlettel,

2. az egyes modulok továbbra is kompatibilisek maradjanak az egyes modulok belso muködésére specifikált - az OMAC API-ban leírt - véges automata modellekkel,

3. a modulok egymásra mutató referenciái megmaradjanak (pl. az *IAxis* tengely interfésztol le lehessen kérdezni a hozzátartozó *IAxisJogging* tengely-jog interfész elérhetoségét).

Hasonló megoldás az MMS (Manufacturing Message Specification) szabvány [41] 88 szolgáltatása és objektuma tekintetében is elfogadott és muködoképesnek bizonyult. Így sokkal reálisabb és olcsóbb egy OMAC vezérlo kifejlesztése.

A továbbiakban részletesen bemutatom a tengelyvezérlo modulra kidolgozott egyszerusítést. (A Függelék 11.3 és 11.4 részében az IAxis és IControlLaw modul esetében a '!' karakter jelzi a minimum készletbe bevonni javasolt eljárásokat.)

- A tengelyvezérlo modul esetében egyszerusítheto a specifikált véges automata modell, amely egyértelmu állapotokba sorolja a tengelyt minden idopillanatban. Megoldásnak javaslom a tranziens jellegu állapotok kivévelét az összes állapot közül. Így pl. az eredeti specifikációban az *enableAxis*() hívás hatására a tengely *enabling* állapotba kerül, majd az engedélyezési folyamat lefutása után magától átkerül *enabled* állapotba. A tranziens állapotokat opcionálissá téve, a hozzájuk rendelt külön lekérdezések is opcionálissá lesznek.

- A tengelyvezérlohöz tartozó néhány interfész teljes egészében opcionálissá teheto (*IAxisDisabling*, *IAxisEnabling*, *IAxisResetting*), mert nem feltétlenül szükséges az általuk leírt vezérlési feladatok ilyen mélységu elérése, a tengelyvezérloben meglévo *disableAxis*(), *enableAxis*(), *resetAxis*() hívások elegendoek.

- Más interfészeknél (*IAxisAbsolutePositioning*, *IAxisHoming*, *IAxisJogging*, *IAxisPositioningServo*, *IaxisVelocityServo*, *IAxisIncrementing*, *IAxisTorqueServo*) elegendo az alapfunkciókat megtartani *startXXX()*, *stopXXX()*, *resetXXX()*, *updateXXX()* stb., minden más opcionálissá teheto.

- Opcionálisnak javaslom az *IAxisLimits*, az *IAxisSupervisor*, az *IAxisSetup* és az *IAxisRates* interfészeket, amelyek szolgáltatásai szintén nem feltétlenül szükségesek minden tengelyvezérloben.

- Meg kell tartani az alapkészletben a tengelyvezérlo legfontosabb I/O interfészeit (*IAxisCommandedInput*, *IAxisCommandedOutput*, *IAxisSensedState*), az *IControlLaw*-t és természetesen az *IAxis*-t, ezekben csak egy-egy hívás teheto opcionálissá.

Ezzel az egyszerusítéssel jelentosen lecsökkentettem a tengelyvezérlo modul komplexitását, a különbözo interfészekben a hívások száma 100 alá csökkent (kb. a 80% válik opcionálissá), miközben a modul az OMAC API egyéb eloírásainak megfeleloen programozható marad. A 8.4 alfejezetben bemutatott prototípusban is a fenti elvek szerinti OMAC tengelyvezérlovel dolgoztam.

5 NYÍLT INTERFÉSZEK VEZÉRLÉSEKBEN - EGY ÍVHEGESZTO ROBOT PÉLDÁJA

A 4. fejezet elején (4. táblázat) leírtak alapján vezérlések külso interfészérol, annak nyíltságáról kétféle értelemben lehet beszélni. Ez a rendszerintegráció eggyel magasabb szintjét jelenti a 4. fejezetben tárgyalt belso architektúrákhoz képest. Az egyik szempont az, hogy a vezérlés távolról - tipikusan egy LAN-on keresztül - nyílt és elérheto legyen, ez a nyílt ipari hálózatokat jelenti. A másik terület pedig az, hogy a vezérlésben végrehajtandó program kód legyen nyílt a különbözo rendszerek (vezérlések, off-line programozói munkahelyek, CAD állomások) felé, ami elsosorban adatformátum kérdése. A 90-es évek tapasztalata, hogy mindkét területen nagyon lassan fejlodik a gyártásautomatizálás a nyílt megoldások felé. A ténylegesen nyílt interfészu rendszert felfogásom szerint az jelenti, ha valahol mindkét feltétel adott.

A fejezetben egy ívhegeszto robottal kapcsolatos ESPRIT kutatásban elért eredményeim adnak erre példát.

5.1 AZ MMS NYÍLT IPARI ADATHÁLÓZAT SZABVÁNY

A nyílt ipari hálózatokkal kapcsolatban a 80-as évek elején NC vezérlésgyártók és felhasználók egy csoportja az EIA (Electonic Industries Association) IE31 számú bizottságának pártfogásával kidolgozott egy ajánlást (#1393A) gyártórendszerek belso adatátviteli problémára. Ez lett az alapja a General Motors vezetésével a MAP (Manufacturing Automation Protocol) megalkotása során kidolgozott általánosabb üzenetkezelo rendszernek, az MMFS-nek (Manufacturing Message Format Standard). Az MMFS a MAP V2.0-ban jelent meg 1984-ben, és segítséget nyújtott NC, PLC, robot és egyéb berendezések kommunikációjához. Ennek hibáit (több interfész inkompatibilitás) küszöbölte ki az ISO 184-es számú technikai albizottsága és így került be a MAP V3.0-ba 1988 novemberében. Az MMS (Manufacturing Message Specification) [41] a MAP-nek az egyik alkalmazói (7.) rétege, egyben az egyetlen olyan része, amely konkrétan gyártásautomatizálás specifikus. Így a MAP késobbi kudarca nem szüntette meg az érdeklodést az MMS iránt, elterjedt nem csak MAP, vagyis OSI, hanem TCP/IP hálózat felett is, valamint elkészült egy Profibus-os változata is. Ma is sok CNC, PLC és robotvezérlo támogatja (Siemens, Allen-Bradley, Modicon, Fanuc stb.), hiszen nagy ipari rendszerek muködnek ma is MMS alapú ipari hálózattal összekapcsolva (pl. az Opel szentgotthárdi gyára). Napjainkban az MMS az USA-ban a közmuveknél a legelterjedtebb, ahol távoli leolvasást és vezérlést biztosító méro és kapcsoló eszközök kommunikációjában használják széles körben [55].

Az MMS a gyártóeszközökrol egy virtuális képet nyújt, ahogyan egy másik készülék a hálózaton keresztül el tudja érni az adott berendezést. Összesen 9 objektumot és a hozzájuk tartozó mintegy 80 szolgáltatást definiál a szabvány. Ezekbol építheto fel a berendezésnek a hálózatról elérheto adatmodellje. Az MMS szabványt egészítik ki az ún. MMS társszabványok, amelyek tipikus gyártóeszközök (robot, NC stb.) MMS-beli leírását adják meg. Sajnos ezeknek a szabványoknak a kidolgozottsága és így a használhatósága is elég gyenge színvonalú.

Minden egyes objektumhoz különbözo számú szerviz tartozik, valamint van néhány általános - a hálózati kapcsolatot kezelo - szerviz is (kapcsolat felvétel, kapcsolat lebontás, abort, kapcsolat visszautasítás stb.).:

MMS objektum neve		fontosabb szervizei
Virtual Manufacturing Device	virtuális gyártóberendezés	identify, status, get name list
Domain	tartomány, eroforrás	download, upload, create, del.
Program Invocation	program invokáció	start, stop, reset, resume, kill
Variable (+ List)	változó	read, write, report
Type	típus	define, delete, get attribute,
Event (Condition, Action Enrollment)	esemény	obtain, notify, define, delete,
Semaphore	szemafor	take control, define, delete,
Operator Station	operátor (monitor + billentyu)	input, output
Journal	napló	create, read, delete, clear

11. táblázat Az MMS objektumai és azok hívásai

Mivel a nyílt ipari hálózatok, ill. ezen belül az MMS specifikáció széles körben ismertek, azért ezek ennél részletesebb bemutatásától itt eltekintek.

5.2 NYÍLT ADAT INTERFÉSZEK IPARI VEZÉRLÉSEK SZÁMÁRA

Az NC és robot technikában hatalmas fejlodés zajlott le az elmúlt évtizedekben, ugyanakkor a szerszámpálya generálás és a szerszámpálya vezérlés különálló feladat maradt lényegében a mai napig. A szerszámpályát a CAD és egyéb információk alapján a különbözo rendszerek off-line generálják, és így állítják elo az ún. G kódokat az NC-k esetében. Majd ezt hajtja végre a CNC rendszer valós idoben, úgy, hogy tipikusan két paraméter változtatására nyílik lehetoség on-line: ezek az elotolás és a foorsó forgási sebessége. Robotoknál nincs egységes robotprogram nyelv, de a meglévok nagy része - a G kódokhoz hasonlóan - a számítógépes assembly nyelveknek feleltetheto meg.

Alapveto probléma ezzel a megoldással, hogy sok információ elvész, amely még megvolt a CAD modellben, ill. a kiegészíto technológiai adatokban. Ezáltal az on-line beavatkozások köre, amelyre szerszámtörés, minoségbiztosítás vagy bármi más miatt szükség lehet, nagymértékben lecsökken. Meggyozodésem, hogy ez a jelenleg használt NC programozás az új generációs (nyílt és intelligens) CNC vezérlések kialakulásának egyik lényegi akadálya.

A cél az, hogy olyan leíró programnyelv keletkezzék, amely a pályagenerálás egy részét az NC ill. robotvezérlo valós idoben végezze el, mert így van komolyabb remény a vezérlo szintjén az esetleges intelligens beavatkozásra. Nyilvánvaló, hogy az eredeti CAD-ban meglévo pályaleírás nem használható változtatás nélkül, hiszen komoly számítási eroforrás és foképpen idobeli kötöttségek vannak, hiszen egy adott pályaelem számítására mindössze az elotte levo pályaelem végrehajtási ideje áll rendelkezésre. Ez a feltétel kicsit enyhítheto bizonyos eloreszámítással, ugyanakkor bármilyen üzemi paraméter (pl. az elotolás) változtatása azonnali változást, azaz újraszámítást jelent.

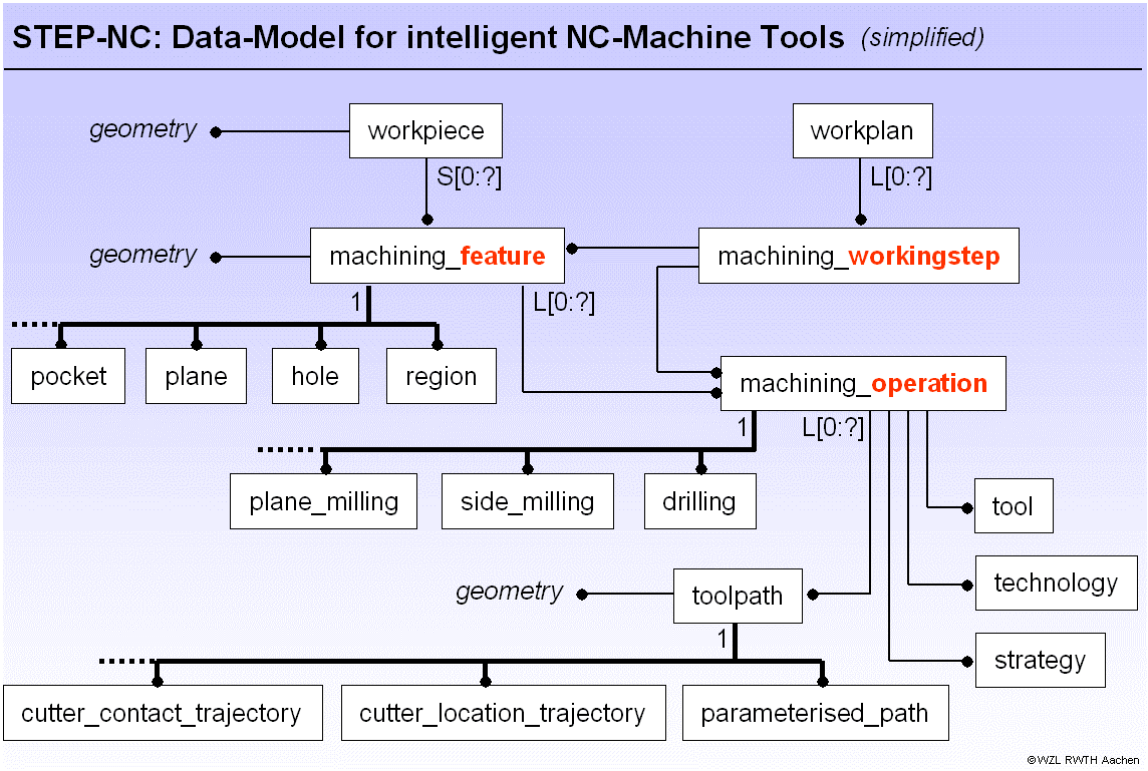
Nemzetközileg is újszeru magyar eredmény volt, hogy a SzTAKI-ban fejlesztett Dialog CNC B-spline opciójával futási idoben lehetett pályát generálni a tartópontok alapján [117].

Ebben az irányban a jelenleg futó legkomolyabb kutatás az ún. STEP-NC [162] nevu IMS projekt, de más hasonló, ám független kutatások is ismertek (pl. Zhang és társai [178],vagy az OSEL [134]).

A STEP-NC célkituzése, hogy teljesen kiváltsa a G kódokat egy STEP alapú termékleírással, amelyet azután a CNC-k végre tudnak hajtani. A másik oldalról olyan leírást ad, amely egységes és minden CAD rendszer támogatja, ill. képes ebbe a formátumba konvertálni a saját termékleírását. A STEP-NC fájl struktúrája a következo:

- projekt fájl
- foprogram (a végrehajtandó megmunkálási feladatok szöveges leírása)
- technológiai leírás (a megmunkálási feladatok objektum orientált leírása)
- geometriai leírás (a geometria STEP alapú leírása)

Mivel olyan nagy az igény egy ilyen jellegu megoldásra a STEP NC már a szabványosítás elso lépéseinél tart [87]. A STEP-NC ugyanakkor meglehetosen bonyolult leírás, összesen több, mint 150 adattípust (application object) definiál. Hátránya, hogy ezzel megszunik az "olvashatóság", hiszen a G kódú programot egy CNC kezelo viszonylag jól megérthette.



13. ábra A STEP-NC jelentosen egyszerusített adat modellje

Zhang megoldásában egy sokkal egyszerubb szintén STEP alapú leírás generálódik off-line, az ún. NCFU (NC feature unit) alapú leírás. Ez a leírás három típusú NCFU alapelem listájából áll, melyek a következok: 1, kontúr-párhuzamos; 2, irány-

párhuzamos; 3, hálószeru felület. Ezek alapján on-line generálódik felület interpolálással a szerszám pálya, majd pálya interpolálással generálja a rendszer ezen elemi elmozdulás-vektorok sorozatát.

Az OSEL háromféle leírást ad az NC programról, az elso az OSEL-F (Feature), amely Java alapú feature alapú leírás. A másik kettorol - az OSEL-I (Intermediate) és OSEL-X (Executable) - nincs publikus részletes leírás.

A fenti munkák természetesen nem elozmény nélküliek. A STEP-NC közvetlen elozményének tekintheto az ESPRIT OPTIMAL [131] projekt, amely a CAM és CNC rendszerek közötti hatékonyabb gyártási információ definiálását és az ezt támogató szoftverek kifejlesztését tuzte ki célul maga elé.

A robottechnikában pedig évek óta az jelent problémát, hogy szemben az CNC-kkel és a G kóddal, az ipari robotok programozására nincsen egyértelmuen kialakult szabványos robot programozási nyelv, ill. nagyon is sokféle van. A CAD és off-line programozói eszközök, amelyek igyekeznek minél többféle robotot támogatni, kényszeruen 3-10 opcionális ún. CAM kimenettel rendelkeznek, azaz ennyiféle robotprogram nyelven képesek kódot generálni. Az alábbi táblázat a legfontosabbakat mutatja be és érzékelteti, hogy gyakorlatilag ahány robotgyártó, annyi robot programozási nyelv létezik:

Robotprogram nyelv	Robot / Vezérlo gyártó
IRL	DIN 66312, német szabvány
V+	Adept
KRL	Kuka
RAPID	ABB
MRL	Mitsubishi
MELFA Basic III	Mitsubishi
BAPS	Bosch
SRPL	Siemens
ROBOTStar IV	Reis

12. táblázat Robotprogram nyelvek

Sokféle erofeszítés van évek óta, hogy ez az áldatlan helyzet megszunjék. Számítógépes nyelvek tervezési elveit követve az ABB RAPID robotnyelvét igyekeztek korszerusíteni [38], hogy a nyelv maga jobban támogassa a robot programozás tipikus problémáit.

A STEP szerepe a robotoknál is elotérbe került a 90-es évek második felében. Kifejezetten nagy pontosságú, nagyméretu varratok (hajók) hegesztésénél az InterRob projektben is használták [113].

Az OPTIMAL projekttel egyidoben egy másik ESPRIT kutatásban kis és közepes szériába gyártott, egyszeru geometriájú, ívhegesztett alkatrészekhez off-line robotprogramozási eszköz fejlesztése is folyt a PROARC projektben [27], ahol az általam kifejlesztett robotprogram leírás az OPTIMAL elveit igazolta egy konkrét esetben. A fejezet további része bemutatja a PROARC projektet és azokat a nyílt interfészeket, amelyeket e projekt kapcsán dolgoztam ki.

5.3 NYÍLT KÜLSO INTERFÉSZEKKEL RENDELKEZO ÍVHEGESZTO ROBOT

5.3.1 A PROARC PROJEKT BEMUTATÁSA

Az Európai Közösség COPERNICUS programjában (7831) ívhegeszto robotok számítógéppel segített programozását támogató rendszert (CAD-Based **Pro**gramming System for **Arc** Welding Robots - PROARC) fejlesztett ki az RWTH Aachen (WZL), a Veszprémi Egyetem (Automatizálási Tanszék) és az MTA SzTAKI (CIM Kutatólabor) 1994 és 1996 között [28, 34].

A rendszer fo célja robotos ívhegesztés támogatása elsosorban kis és közepes vállatok (SME) részére, egyedi és kis sorozatnagyság mellett, hatékony programozási módszereket kínálva a hegesztésben járatlanabb felhasználóknak is. Egy ilyen rendszer megszüntetheti, de legalábbis radikálisan csökkenti azt a hosszadalmas idot, ami az új munkadarab betanításából adódik, amikor az operátor kézzel betanította a robotnak az ívhegeszto programot. A PROARC rendszer közvetlenül a CAD adatból olyan ívhegesztési programot generál, ami azonnal, vagy minimális kézi módosításokkal használható. Olyan vállalatoknál, ahol kisebb szériák vannak, ez ugrásszeruen növelheti a termelékenységet. A világban leginkább elterjedt PC alapú CAD rendszerben, az AutoCAD-ben került implementálásra és olyan lehetoségeket teremtett, melyek akkoriban hagyományosan csak nagy munkaállomásokon voltak elérhetok (pl. a legismertebbek és azóta is sokfelé használt ilyen rendszerek a ROBCAD [167] és az IGRIP [49]).

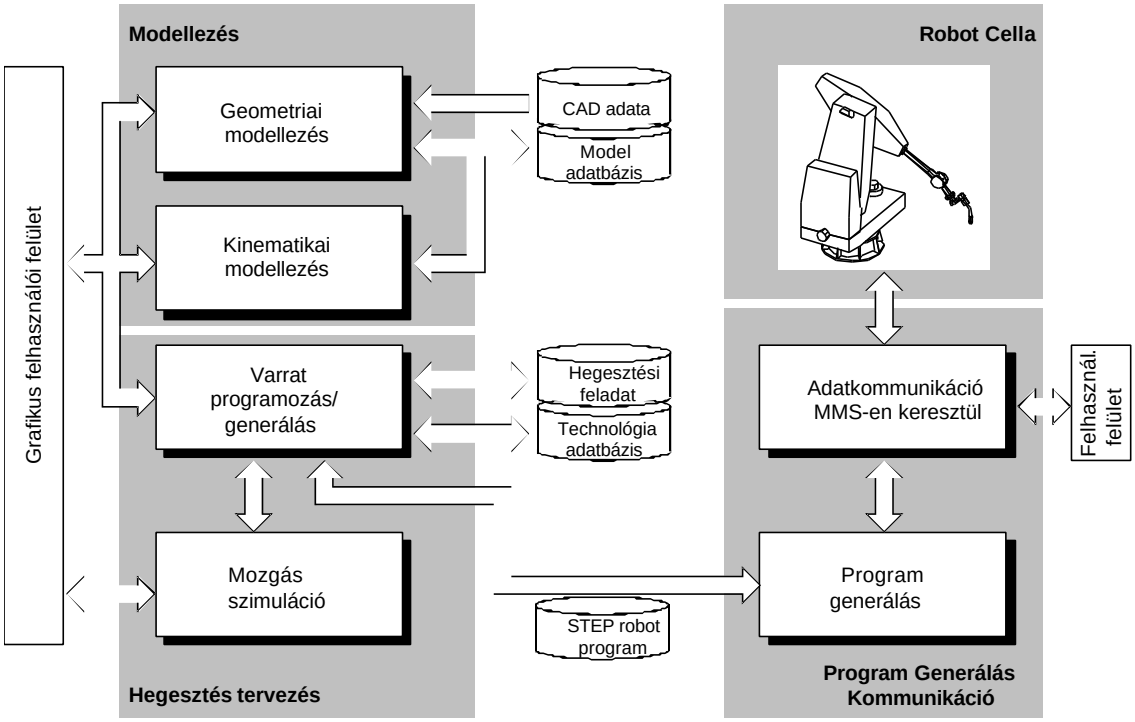
A rendszer egy olyan interaktív off-line robotprogramozási rendszer, amely segít az ívhegesztés egyszeru tervezésében és teljes vagy részleges végrehajtásában egyaránt, kiegészítve szimulációs és automatikus programgenerálási funkciókkal is. A tipikus robotcella felépítését követve egy ívhegeszto robotból és egy (opcionális) pozícionálóból áll, de az esetlegesen meglévo szerszámtisztítót és/vagy szerszámcserélot is támogatja. A munkadarabokat egy vagy többrétegu varratokkal lehet összehegeszteni, a rendszer K és V típusú varratok lineáris vagy körpályán történo hegesztését támogatja. A szükséges hegesztési paraméterek egy technológiai adatbázisból nyerhetok ill. kézzel is beállíthatók.

A felhasználó saját munkadarabjainak terveit elkészítheti vagy betöltheti abba az AutoCAD környezetbe, amelyben a PROARC eszközkészlete is elérheto. Jelenti ez magát az ívhegeszto robot cellát, amely természetesen tartalmazza az adott robot és környezete AutoCAD-beli rajzát, valamint a robot inverz kinematikáját. Ezen kívül a PROARC rendszer egyedi funkcióit, amelyek teljesen beleintegrálódnak az AutoCAD megszokott felhasználói felületébe.

A használat során a felhasználó kijelölheti, hogy mely érintkezo éle(ke)t kívánja összehegeszteni. Ekkor a rendszer a kijelölt él(ek) alapján elemi hegesztési lépésekbol automatikusan felépíti a varratot, emellett pedig az adatbázisából technológiai paramétereket javasol (hány rétegu legyen a varrat, javasolt hegesztési feszültség stb.), amelyeket természetesen felülbírálhat a felhasználó. Ha elkészült az összes varrat kijelölése, akkor a PROARC legenerálja a teljes hegesztési robotprogramot, melynek során ellenorzi, hogy nincs-e ütközés ill., hogy a program megfelel-e az adott robot

kinematikájának. Lehetoség van az AutoCAD-be épített szimulációban végigkövetni az ívhegeszto robot leendo pályáját.

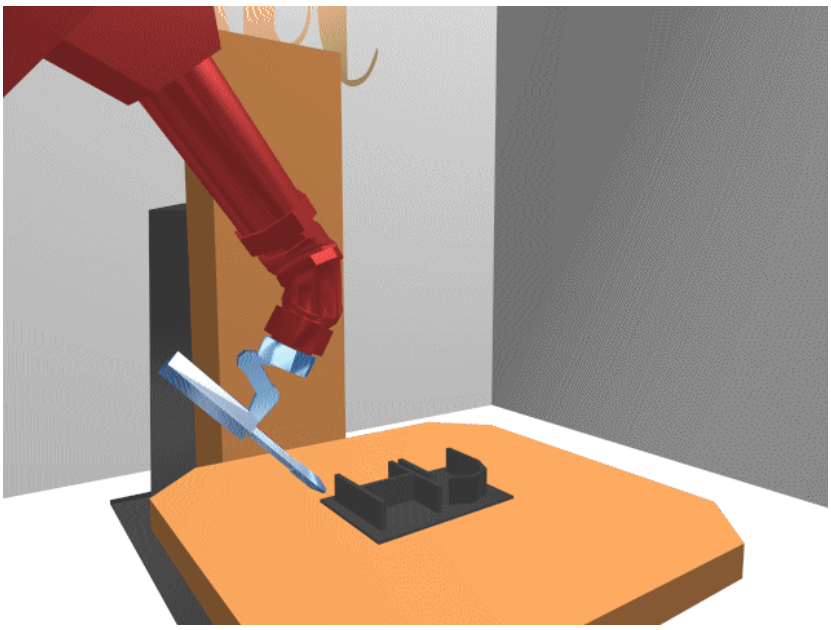
Szabványos, nyílt ipari interfészek segítségével kapcsolódik a rendszer a különféle robotokhoz és hegeszto berendezésekhez. Egyrészt a programozási rendszer egy STEP alapú nyílt ívhegesztési robotprogramot generál, másrészt szabványos MMS alapú hálózati felületen keresztül kapcsolódhat a programozói rendszer az ívhegeszto robothoz. Ehhez definiálnom kellett az MMS és robot szabványokhoz illeszkedo "ívhegeszto robot virtuális megmunkáló eszközt" (AWR-VMD). Egyedül állónak tekintheto, hogy a PROARC projektben mind az adat, mind a hálózati interfész nyílt.



14. ábra A PROARC rendszer logikai felépítése

A modellezés és a hegesztés tervezés szervesen egybeépült és az AutoCAD rendszerbe lett mélyen beintegrálva. Az ívhegeszto robot felé az interfészt a STEP alapú robotprogramom jelenti. Az általam fejlesztett programgenerálás és kommunikáció önálló alkalmazások, a program generáló segédprogram a STEP alapú robotprogramból generálja azt a kódot, amit a konkrét ívhegeszto robot használ, majd a kommunikációs segédprogram a nyílt ipari hálózaton keresztül vezérli az ívhegeszto robotot (letölti a generált robotprogramot, áttölti és elmenti a robotban meglévo robotprogramot, indítja, megállítja távolról a robot muködését).

A PROARC rendszer a Kandó Kálmán Muszaki Foiskolán és az Aacheni Muszaki Egyetemen lett installálva, ahol később az oktatásba is bemutatták a rendszert. A PROARC rendszer mindkét helyen KUKA robot [98] számára generál programot, bár a két helyen más a robotok típusa és kinematikája. A Kandó Foiskolán a robothoz egy ESS [56] gyártmányú ívhegeszto berendezés kapcsolódik, amelynek programozása a KUKA robot I/O portjain keresztül lehetséges. A rendszer sikeresen hegesztett egyenest és körívet, egy- és többszörös varrattal. Hegesztés nélküli kereséssel ellenorizte a munkadarabot.



15. ábra Az AutoCAD szimuláció egy pillanatfelvétele a Kandó bemutatón szerepelt hegesztett munkadarabbal (HU)

5.3.2 STEP ALAPÚ ÍVHEGESZTO ROBOTPROGRAM LEÍRÁS

A PROARC projektben használt STEP alapú robotprogram [16, 29] kialakításánál az alábbi szempontokat vettem figyelembe:

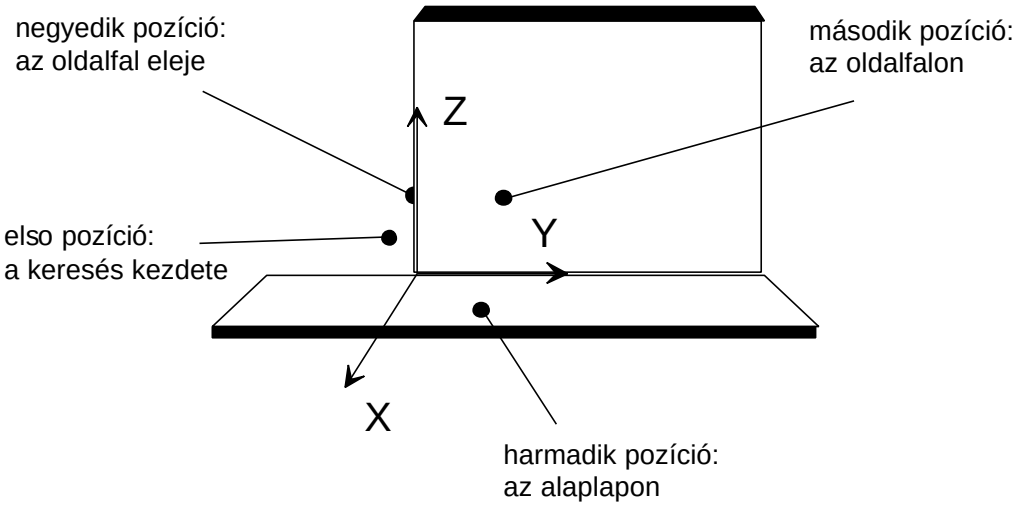
- a konkrét robottól független legyen a leírás és teljes mértékben megfeleljen az ISO 10303-21:1994 szabványnak,
- olyan objektum orientált leírás keletkezzék, mely egyben "olvasható" is marad,
- bovítheto legyen,
- legyen univerzális, vagyis használható legyen olyan robot esetében, amely támogat speciális ívhegeszto funkciókat és olyannál is, amelyik nem,
- támogassa az ívhegesztés tipikus problémáit, így a többrétegu varratok hegesztését, és a hegesztés elején, közben és végén szükséges speciális funkciókat (pl. rezegtetés),
- támogassa érintés és ívszenzor használatát.

Éppen ezért a robotprogram struktúráját (17. ábra) nagyon egyszerunek alakítottam ki, amivel sikerült megorizni a szöveg közvetlen olvashatóságát: A program 4 féle muvelet (*operation*) tetszoleges számú és sorrendu sorozatából áll (*process_sequence*), amelyek a következok lehetnek:

- Pozicionálás (*positioning*), ami elemi mozgások (*move*) sorozata. Egy elemi mozgás leírása az interpoláció típusából (ptp, lineáris, kör), a cél pozícióból (*pose*), a célban a csuklók konfigurációjából (*joint*) és a sebességbol (*speed*) áll. A körmozgás pontos megadásához a körpálya egy közbenso pontja is meg van adva.

- Hegesztés nélküli keresés (*offline_search*), amelyet a varrat azonosító (*seam_group_id*), a keresés geometriai típusa és a keresési szekvenciát leíró keresési pontok (*search_point*) listája definiál. Az egyes keresési pontok leírását (1) az ideális pont pozíciója, (2) a megengedett tolerancia és (3) a visszalépési pozíció - ahova az adott elemi keresés végén kell visszaállni a robotnak - segítségével oldottam meg. A keresés ilyen definíciója biztosítja, hogy a programleírás használható olyan ívhegeszto robotok számára, melyek "beépítetten" (külön utasítással) tudnak keresni és olyanok számára is, ahol a keresési mozgást is részletesen meg kell adni pozícióról pozícióra (mert pl. a hegesztopisztoly és a szenzor kiegészítoként lett felszerelve).

  A hegesztés nélküli keresésben a V és K varratoknál szokásos kereséseket szerepeltetem a keresés geometriai típusai között. Ezek a következok:
  - K varratokra:
    - sarok keresés (*CS*),
    - oldalfal keresés (*SSW*),
    - alaplap keresés (*SSP*),
    - középso keresés (*MS*), ami oldal és alaplap keresés egymás után,
  - V varratokra:
    - jobb oldali keresés (*SBR*),
    - bal oldali keresés (*SBL*),
    - jobb oldali keresés keskeny rés mellett (*SBSR*),
    - bal oldali keresés keskeny rés mellett (*SBSL*),
  - és kikapcsolható a keresés:
    - nincs keresés (*NS*).



16. ábra Az oldalfal keresés (SSW) muveletben definiált pontok

- Hegesztés (*welding*), amelyet a varrat azonosító (*seam_group_id*), a hegesztési pontok (*welding_point*) listája és az egész varratra vonatkozó technológiai paraméterek (*seam_paramlist*) határoznak meg. A hegesztés vége lehet egy sarokban vagy szabad térben, aminek megfeleloen kell a hegesztés közbeni opcionális keresést is beállítani. Ha sarokban van a varrat vége, akkor ugyanazt a szenzort és változókat lehet használni, mint a hegesztés nélküli keresésnél, míg a szabad végu varratnál ívszenzorra van szükség.

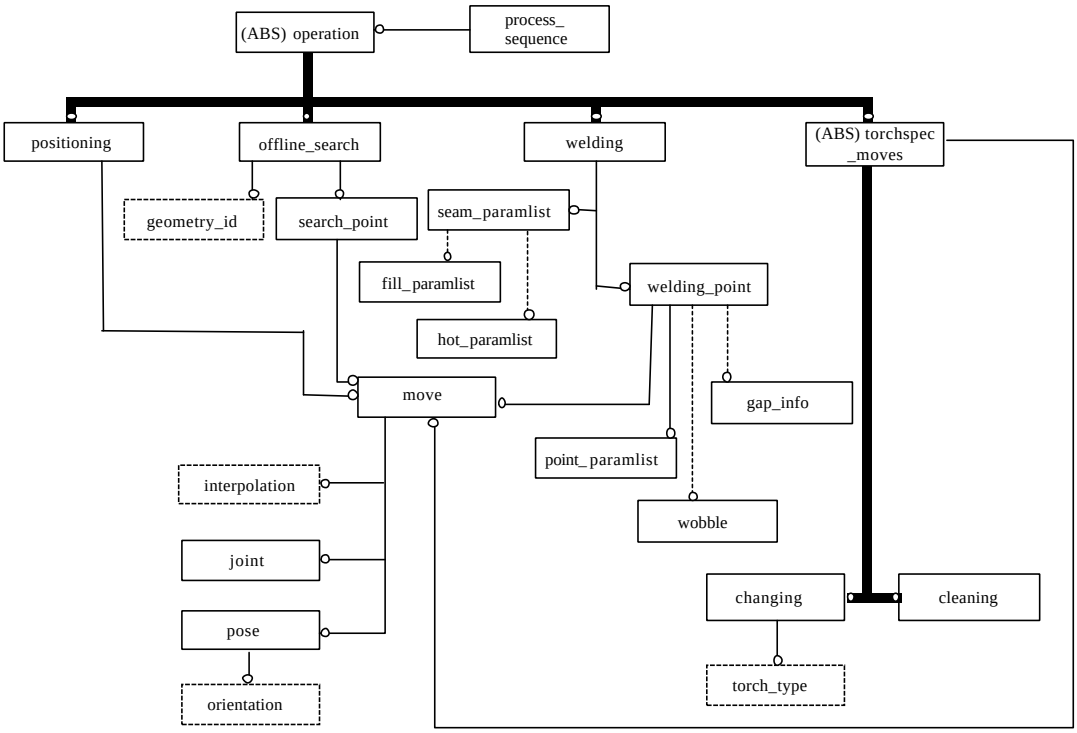
A hegesztési ponttal írom le egy elemi varratdarab hegesztésének körülményeit, amelyek a következok:

- elemi mozgás (*move*),
- rezegtetés (*wobble*) (igen/nem) ill. ennek opcionális paraméterei (amplitúdó, frekvencia, stb.),
- hegesztési paraméterek (*point_paramlist*), vagyis a hegesztési feszültség, a hegesztési alapáram és a huzal sebesség,
- ívszenzor (*arcsensor*) (igen/nem) ill. ennek opcionális paraméterei: a rés típusa, szélessége és szöge,
- sarokkeresésnél a keresési tolerancia és a visszalépési pozíció.

Az egész varratra vonatkozó technológiai paraméterek pedig a következok:

- gázkapcsolási ido a hegesztés elott és után, amely megadja, hogy a hegesztés elott és után mennyivel kell be- ill. kikapcsolni a hegesztési gázt,
- *hot_paramlist*: a hegesztés helyes kezdéséhez mennyivel több (nagyobb sebességu) huzal kell és a hegesztési varrat milyen hosszán,
- *fill-paramlist:* a hegesztés végén hogyan kell a keletkezendo hegesztési krátert plusz anyaggal feltölteni egy visszafelé irányuló mozgással.

- Specifikus mozgások, amelyek az ívhegesztés kiszolgáló feladatait reprezentálják, a pisztoly tisztítását (*cleaning*) és a pisztoly cseréjét (*changing*) jelentik, mely utóbbinak paramétere a jelenlegi és a következo pisztoly típusa. Mindkét speciális feladathoz természetesen mozgássorozat is tartozik.



17. ábra A specifikált ívhegeszto robotprogram EXPRESSG leírása

Mivel egy adott hegesztés nélküli keresés és hegesztés muvelete összekapcsolható a varratazonosító (*seam_group_id*) segítségével, így többrétegu varrat esetében egyetlen kereséshez több hegesztés tartozik.

Az így definiált robotprogrammal lehetoséget teremtettem a keresés eredményeképpen a befogás on-line kompenzációjára is transzverzális hiba esetén.

Természetesen a fenti nyílt ívhegeszto robotprogram leírás magán viseli a PROARC projekt korlátozásait, vagyis nem alkalmas bármilyen ívhegesztési feladat esetében a robotprogram leírására, de a STEP kínálta objektum orientált felépítés és az a mód, ahogyan az entitásokat felépítettem, biztosítja a robotprogram koherens bovíthetoségét. A Függelékben (11.5) található egyszeru példában a rövid STEP robotprogram egy pozícionálást, egy hegesztés nélküli sarok-keresést és egy hegesztést tartalmaz.

A PROARC lezárása óta eltelt idoszak azt mutatja, hogy egyre inkább teret nyernek a CAD rendszerekhez "közeli" robotnyelvek. A STEP használata, mint a PROARC egy újdonsága is visszaköszön más projektekben [145]

A STEP-NC pedig, melynek az OPTIMA eredményei kapcsán a PROARC is egyfajta elozménye, reális lehetoséget teremt arra napjainkban, hogy lecserélhetové váljon az NC technikában a G-kód egy sokkal korszerubb, és az intelligens CNC-k számára szükségszeru programozási nyelvvel.

Ugyancsak a PROARC lezárása óta publikálták a termék és folyamat modellezés szorosabbá tétele kapcsán a STEP használatát hegesztési és CAD adatok összekapcsolására a CAPABLE/Welding rendszerben [110].

5.3.3 ÍVHEGESZTO ROBOT NYÍLT HÁLÓZATBA KAPCSOLVA

Off-line programozói munkaállomások tipikusan nem képesek integrált módon a generált program letöltését ill. futtatását is támogatni. A PROARC-ban viszont cél volt a felhasználó teljes köru kiszolgálása, ezért ezeket a feladatokat is meg kellett oldani [26]. A sokféle egyedi DNC kapcsolat helyett valamilyen nyílt megoldást választva az MMS kommunikációra esett a választás. Ehhez azonban meg kellett vizsgálnom, hogyan lehet egy általános ívhegeszto robotot MMS hálózatba kapcsolni.

Ahhoz, hogy nyílt MMS hálózatba kapcsolható legyen egy ívhegeszto robot, eloször az MMS szabványnak (ISO/IEC 9506-1 és 2) megfelelo "ívhegeszto robot virtuális megmunkáló eszközt" AWR-VMD (ArcWelding Robot VMD) [14, 29] kellett definiálnom.

A kiindulást számomra Nagy Gergely és Haidegger Géza által a SzTAKI-ban korábban kifejlesztett általános RobotVMD biztosította [125]. Az MMS robot társszabványának (ISO/IEC 9506-3) egy szukítését és egyfajta logikai letisztítását jelenti a RobotVMD, amely csak azokat az objektumokat tartalmazza, amelyek tipikusan elérhetok egy hagyományos robotvezérloben a külso (DNC) kapcsolat segítségével.

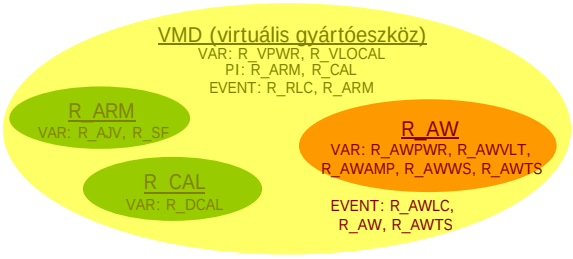
A 13. táblázat a robot társszabvány azon objektumait tartalmazza, amelyeket a RobotVMD megorzött, és amelyekkel mintegy 8 különbözo gyártmányú robotot lehetett az évek során nyílt MMS hálózatba kapcsolni [73].

Név	MMS objektum típus	Jelentés	Megjegyzés, attribútumok
	VMD (virtuális gyártóeszköz)		Vendor: "SZTAKI", Modell: "Robot Controller", Revision: "2.0"
R_VPWR	Variable (logikai)	Hajtás bekapcsolva	
R_VLOCAL	Variable (logikai)	Lokális vezérlés	
R_ARM	Domain (tartomány)	A robot programja	
R_ARM	Program Invocations (program muködtetés)	Valamely kiválasztott robot-program futtatásának vezérlése	
R_AJV	Variable (6 elemu valós tömb)	A robot pozíciója és orientációja	R_ARM domain specifikus
R_SF	Variable (valós)	A robot sebessége	R_ARM domain specifikus
R_CAL	Domain	A robot kalibráció adatai	
R_CAL	Program Invocation	A robot kalibráció vezérlése	
R_DCAL	Variable (8 bites integer)	Robot kalibrálva	R_CAL domain specifikus
R_RLC	Event (esemény)	Lokális/távoli vezérlésben változás történt	
R_ARM	Event	Az R_ARM PI elhagyta a "fut" állapotot	

13. táblázat Robot VMD objektumai

Az MMS robotszabvány lehetové teszi ilyen speciális célú VMD definiálását azzal, hogy lehetséges a szabványon túlmeno objektumokat is definiálni, de konkrét megoldást nem ad. Alapveto fontosságúnak tartottam a meglévo RobotVMD-vel megorizni a teljes azonosságot, hogy az AWR-VMD mindenben kompatibilis legyen egy egyszeru RobotVMD-vel.

Az AWR-VMD kialakítása során felmerült bennem, hogy a hegeszto berendezés közvetlenül kapcsolódjon-e az ipari hálózathoz vagy pedig szorosan a robothoz kapcsolódó külso egységnek tekintendo, hiszen az MMS szabvány erre is lehetoséget teremt. Tehát egy önálló VMD-t kell-e kialakítani vagy pedig a meglévo RobotVMD kapjon egy újabb - az ívhegesztéshez kapcsolható - tartományt (domain-t). A második megoldás mellett döntöttem, mivel a demonstrációs konfiguráció az utóbbit támogatta, és az olcsóbb kategóriájú ívhegeszto robotoknál is ez a fajta integrált kivitel mondható tipikusnak.



18. ábra AWR-VMD tartományai

Ugyanakkor a kidolgozott specifikáció kompatibilis az olyan környezetekkel is, amelyben drágább és önállóan is vezérelheto hegeszto berendezés van. Ekkor az R_AW tartomány elemeit egy önálló hegeszto-berendezés VMD-be kell felépíteni, a táblázatban megadott R_AW elemeket pedig változatlanul kell hagyni. Természetesen ez a VMD csak az egyetlen R_AW tartományt tartalmazza.

A 14. táblázat tehát RobotVMD leírás bovítésével létrejött AWR-VMD specifikációm objektumait tartalmazza, amely az elso - és tudtommal egyetlen - olyan MMS alapú ívhegeszto robot VMD specifikáció, amelyik ténylegesen muködött:

Név	MMS objektum típus	Jelentés	Megjegyzés, attribútumok
R_AW	Domain (tartomány)	Az ívhegeszto adatai	
R_AWPWR	Variable (logikai)	Ívhegeszton bekapcsolt feszültség	R_ARCW domain specifikus
R_AWLOC	Variable (logikai)	Ívhegeszto lokális vezérlése	R_ARCW domain specifikus
R_AWVLT	Variable (valós)	Hegesztési feszültség	R_ARCW domain specifikus
R_AWAMP	Variable (valós)	Hegesztési áram	R_ARCW domain specifikus
R_AWWS	Variable (valós)	Huzal sebesség	R_ARCW domain specifikus
R_AWLC	Event (esemény)	Az ívhegesztoben lokális/távoli vezérlésben változás történt	
R_AW	Event (esemény)	Az ívhegesztoben a hegesztési feszültség ki/be kapcsolásban változás történt	
R_AWTS	Variable (logikai)	Érintés szenzor	R_ARCW domain specifikus
R_AWTS	Event (esemény)	Érintés szenzor értékében változás történt	

14. táblázat AWR-VMD bovítései

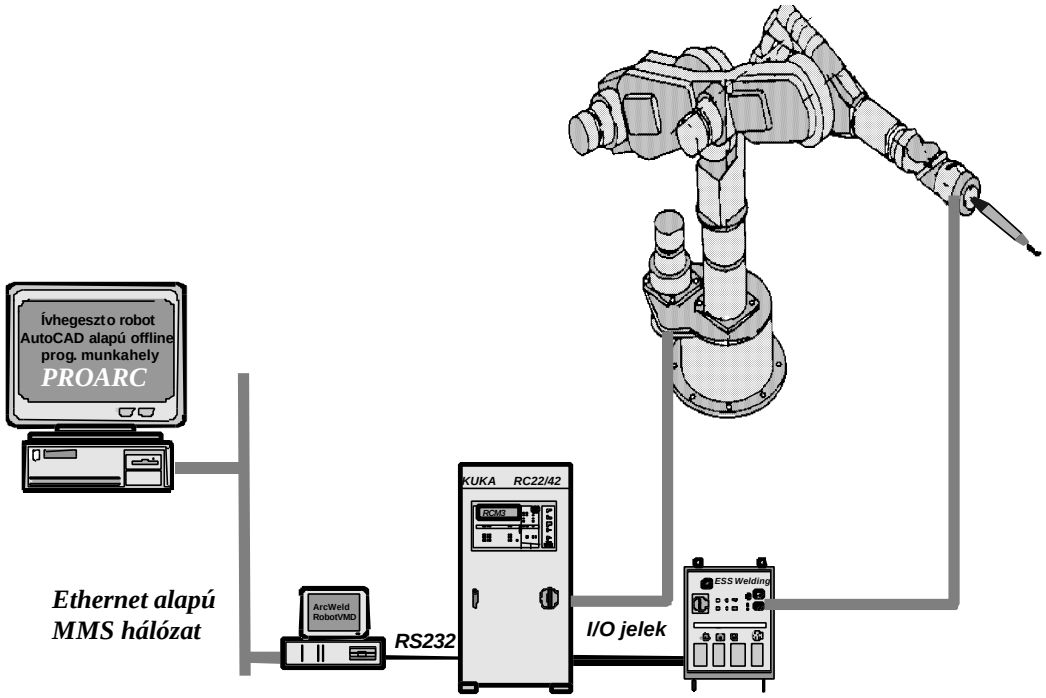
Az alábbi felsorolás azokat az MMS szervizeket tartalmazza, amelyeket sikerrel használtak a kifejlesztett RobotVMD-k, így az AWR-VMD is. Ez a halmaz a szabvány MMS funkciók közel 80 %-át jelenti :

Context Management (kapcsolat kezelés):	initiate, abort, reject, conclude, cancel,
VMD Support:	status, unsolicited status, get name list, identify, get capability list,
Variable:	read, write, information report, get variable access attributes,
Domain:	initiate download, download, terminate download, initiate upload, upload, terminate upload, get domain attributes,
Program Invocation:	create, delete, start, stop, reset, resume, get pi attributes,
Event:	notification, acknowledge, alter event condition monitoring

Konkrétan a PROARC projektben az AutoCAD-es (Pentium, Windows 95) tervezoi munkahely a nyílt hálózaton egy ún. hálózati interfész egységhez (NIU - Network Interface Unit) (i486, OS/2) kapcsolódott. Mindkét gépen a SISCO MMS-EASE szoftverén [159] alapuló MMS szoftverem futott. Az NIU-n AWR-VMD, mint MMS szerver, míg a PROARC tervezoi gépen egy MMS hálózati kliens. Az NIU soros porton

kommunikált a KUKA robottal (Siemens RCM3-as vezérlo) és annak digitális I/O csatornáin érte el az ESS hegeszto berendezést (19. ábra).

A PROARC projekt bemutatóin az AWR-VMD bovítésnek csak egy részhalmazát tudtam ténylegesen használni, mert nem minden változó elérhetoségét biztosította a KUKA robot hagyományos DNC csatornája.



19. ábra A PROARC budapesti bemutatójának elemei

A PROARC lezárása óta indult és futó kutatásokat vizsgálva, melyek bár sokkal nagyobb ipari és anyagi háttérrel bírnak, ugyanazok az irányok figyelhetok meg. Jelentos a NIST-ben 1996-ban indult AWMS projekt (Automated Welding Manufacturing Systems) [145], melynek egyik fo célkituzése, a szabványos interfészek használata, a PROARC eredményeit igazolja. Ugyanakkor az elmúlt 2-3 év eredménye az, hogy az ipari nyílt hálózatok helyett az Internet technológia felé fordult a figyelem, az ívhegeszto robot cella pedig multimédiás környezetbe került [67].

6 NYÍLT VEZÉRLÉSEKBE INTEGRÁLT INTELLIGENS MEGOLDÁSOK

Az intelligens rendszerek nyitottságának a kérdése hagyományosan a felhasználói felületek nyitottságával kapcsolatban merül fel. Hernandez [75] háromféle rendszert különböztet meg:

- Intelligens Rendszer, ahol az adott MI alkalmazás egy hagyományos felhasználói felületen keresztül kapcsolódik a külvilághoz.

- Intelligens Felhasználói Felület, amikor egy viszonylag zárt, fekete doboz jellegu alkalmazás esetében a felhasználó típusának megfelelo módon generál választ a rendszer.

- Intelligens Interfész, ahol a tudásbázisú architektúra mellett a felhasználótól függoen fog következtetni a rendszer, vagyis a belso muködését határozza meg, hogy milyen típusú felhasználót szolgál ki.

Ebben a fejezetben szereplo eredményeim esetében a nyílt platform teremtette meg a lehetoségét az intelligens alkotóelem viszonylag egyszeru rendszerbe integrálására és így az adott vezérlés intelligenciájának a növelésére.

6.1 NYÍLT HÁLÓZATBAN KIALAKÍTOTT INTELLIGENS CELLAVEZÉRLÉS

A SzTAKI-ban üzemelo MAP Training Center [21] és a különféle, szakérto rendszerrel kapcsolatos szimulációs, ütemezési és minoségbiztosítási kutatások [6] vezettek el engem a nyílt ipari hálózati környezetben megvalósított MI alkalmazásokhoz, amelyek a szimuláció helyett valódi vezérlési funkciókat is ellátnak.

6.1.1 ROBOTCELLA G2 ALAPÚ VEZÉRLÉSE MMS HÁLÓZATBAN

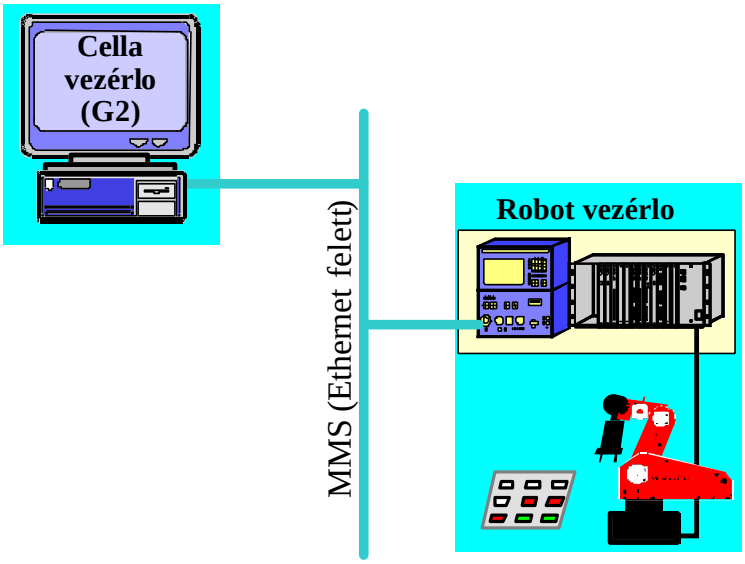
Noha a MAP, mint a gyárinformatika egy fontos eleme, az MI alkalmazásról szóló könyvekben (pl. [100]) is helyet kapott, a MAP nyújtotta nyíltságot sokáig nem használták ki az MI alkalmazásokban. Az elso ilyen ismert prototípus rendszert készítettem el, amikor a SzTAKI-ban egy robot cellát [22] intelligens cellavezérléssel építettem fel MMS hálózatban. A rendszer elkészítésével az volt a célom, hogy a MAP/MMS nyíltságát egy intelligens környezetben tudjam kihasználni, illetve azt akartam igazolnia, hogy a G2 következtetései - a nyílt hálózatba kapcsolt berendezések adatainak on-line elérésével - valódi kontroll funkciót is meg tudnak valósítani.

A G2-re [65] több dolog miatt esett a választásom. A korábban részletezett tulajdonságok mellett (2.3 alfejezet) ugyanis ez az intelligens környezet kifejezetten alkalmas ötletek, prototípusok gyors kipróbálására.

Más kutatóhelyeken is használják a G2-t hasonló célokra, így pl. Szöulban kísérleti célú virtuális gyár muködtetéséhez egy cellavezérlot [101] fejlesztettek ki a G2

intelligens környezetben, mely képes a cellában levo berendezésekkel kommunikálni. Egy rendszerben integráltak dinamikus sorrendtervezot, több lépéses ütemezot és vezérlot, melyek közös adat és tudásbázist használnak.

A kísérleti rendszerem alapját egy Mitsubishi robot képezte, amelyet elore megtanított pozíciókba lehetett vezérelni. A robotvezérlo soros DNC kapcsolatán keresztül egy PC-hez volt kapcsolva, mely egyben az MMS hálózat felé hálózati interfész egységként funkcionált. Ez az egység volt a SzTAKI RobotVMD-jének [125] elso prototípusa. Az MMS hálózatra egy G2 [11] alapú cellavezérlo kapcsolódott, amelyben egy egyszeru példaalkalmazás vezérelte a robotot (20. ábra). A közismert amoba játékot lehetett játszani a géppel úgy, hogy mind a felhasználó, mind a gép lépéseit végrehajtotta a robot. Az MMS szabványnak megfeleloen a robotprogramokat (jelen esetben a lépéshez szükséges pozíciók listáját) mindig letöltötte a cellavezérlo, majd elindította a robot programot. A robot a piros és zöld játékkockákat a tárolóból vette elo és helyezte arra a mezore, amit a játékos vagy a gép kijelölt.

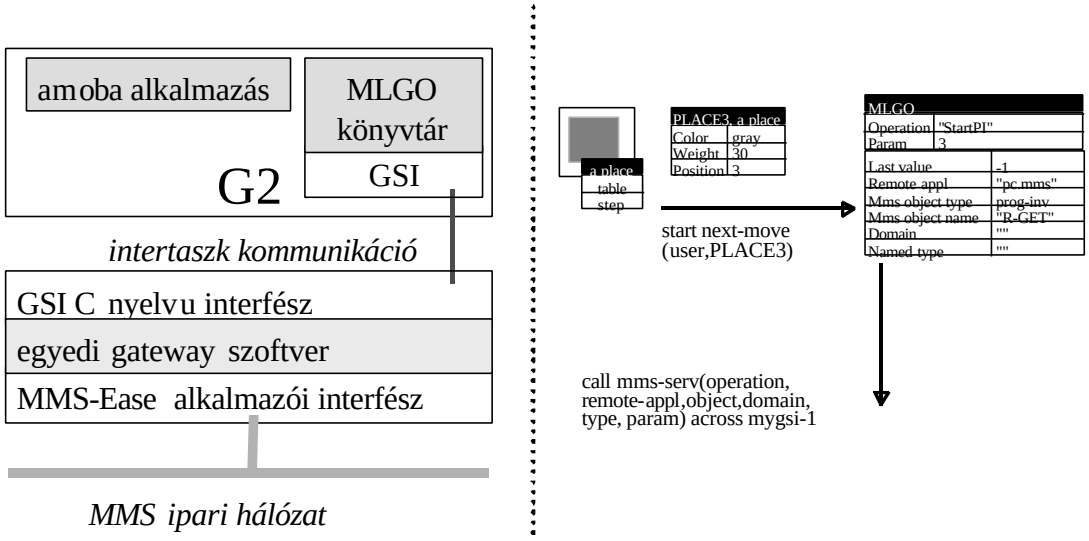


20. ábra Tudásbázisú (G2 alapú) nyílt (MMS) robotos cellavezérlo prototípusának elemei

A tudásbázisú G2 rendszer objektum orientált felépítésére alapozva a vezérlendo virtuális gyártóeszközök olyan reprezentációját dolgoztam ki, amely túl azon, hogy kelloen általános, alkalmas a tényleges hálózati protokollal együttmuködni és a G2 következteto mechanizmusa is hatékonyan használni tudja. A megoldást az jelentette, hogy felismertem, hogy az MMS protokoll objektum orientált szemléletben egy az egyben alkalmas erre a feladatra, ha a G2-ben megfelelo módon reprezentálják. Két megoldás elemzését végeztem el. Eloször a teljes MMS objektum struktúrát (mind a 86 szervizhívással) felépítve G2-ben egy (feleslegesen) bonyolult objektum és metódus halmazt hoztam létre. Ennél sokkal jobb megoldást találtam, amikor - megkeresve a szolgáltatásokban (pl. tartomány, program invokáció, változó) azokat az elemeket, amelyek egységesen paraméterezhetok - egy lényegesen kompaktabb objektum és szerviz halmazt tudtam felépíteni, amelyet MLGO (MMS like G2 Objects) objektum könyvtárnak neveztem el [7].

Az MMS változó objektumában használt akcióknak - létrehoz (*create*), olvas/ír (*read/write*) és töröl (*delete*) - sikerült a program invokáció, a tartomány és az esemény objektum megfelelo akcióit is megfeleltetnem. Az esemény esetében ez értelemszeru, a másik ketto esetében viszont az adott objektumhoz tartozó MMS akciók összevonásával tudtam ezt a megfeleltetést biztosítani. A "létrehoz"-nak megfeleltetett aktivitás töltötte le az aktuális programot a berendezésbe (tartomány létrehozás és letöltés) és helyezte a berendezést üzemkész állapotba (program invokáció létrehozás), majd pedig az "írás" indította a robot futását (start program invokáció). Ez azt jelentette, hogy bizonyos alkalmakkor a tudásbázis egyetlen kezdeményezésére az MLGO egymás után, automatikusan több MMS szervizt hívott meg. Így a cellavezérlo belso következteto mechanizmusa megszabadult attól a tehertol, hogy az MMS logikájából következo részletes állapotátmeneteket külön-külön kelljen meghívnia.

Ezek alapján a G2 alapú cellavezérlot két részbol állítottam össze. A virtuális gyártóeszköz(ök) leírása az MLGO könyvtárbeli objektumokkal történt, és ennek metódusait hívta a konkrét alkalmazás (pl. az amoba), ha a hálózaton keresztül üzenetet akart küldeni a gyártóeszköznek. Az MLGO objektumokat és a hozzájuk tartozó eljárásokat kapcsolta össze a G2 rendszer interfészén keresztül (GSI) egy általam fejlesztett gateway program a SISCO MMS-EASE szoftverével [159], amely a rendszert az MMS hálózatba kapcsolta. Az MLGO objektumrendszerének G2-beli sikeres megalkotásával igazoltam, hogy az MMS, mint objektum orientált hálózati protokoll, egyben alkalmas modellezo eszköz is nyílt vezérlések kapcsolatainak leírására.



21. ábra A G2 alapú cellavezérlo szoftver felépítése és MMS szerviz hívási mechanizmusa

A nyílt hálózatba kapcsolt, intelligens vezérlo prototípusának publikálása [8] után szóba került esetleges részvétel egy nagyszabású amerikai ipari projektben [85], amely végül nem a SzTAKI közremuködésével valósult meg. A Ford egyik üzemében üzembe helyezett rugalmas gyártórendszert G2 alapú cellavezérlo irányítja, mely Fanuc és Allen-Bradley vezérloket, AGV-ket irányít DEC és Oracle környezetben MMS ipari kommunikációs hálózattal összekötve. A G2 határozza meg, hogy az egyes palettákat hova vigye az AGV rendszer (hol optimális az adott megmunkálást elvégezni), tölti le az NC programokat a szerszámgépekbe, indítja és felügyeli a megmunkálást.

Figyelmezteti az operátort, illetve - ha szükséges - ideiglenesen kiveszi a problémás munkaállomást a cellából.

6.1.2 A PROARC RENDSZER TUDÁSBÁZISÚ BOVÍTÉSE

A PROARC rendszer (5.3.1 fejezet) fejlesztése során több olyan feladat merült fel, amelyek túlmutattak a PROARC keretein és felvetették bennem a rendszer egy opcionális intelligens modullal való kiegészítését. A korábban kifejlesztett nyílt robotcellás (6.1.1 alfejezet) eredményem - a G2 rendszer és az MMS nyílt hálózati protokoll összekapcsolása - jelentette az alapot az elképzelésem realizálására [8].

A hegesztés támogatása szakérto rendszerrel nem új gondolat, több sikeres kísérleti rendszer ismert. Viszonylag jó hegesztési paraméterek beállítása ugyanis kifejezetten nehéz probléma ívhegesztés esetén. A különbözo humán szakértok véleménye ill. adott munkadarabok esetén javasolt beállításai is sokszor nagymértékben eltérnek egymástól. A szakértok által jónak tartott - idonként ellentétes - beállítások, szabályok alapján ad konkrét javaslatot a hegesztési paraméterekre egy korai LISP alapú szakérto rendszer [170]. Ennek fejlesztése során azt is megállapították, hogy a szakérto rendszernek figyelembe kell vennie, hogy a hegesztési paramétereket milyen sorrendben specifikálják, mert pusztán a sorrend miatt is komoly eltérések adódtak a humán szakértok között.

Nem ismert azonban olyan tudásbázisú rendszer, amely az ívhegesztési technológiát olyan módon támogatja, hogy annak a teljes folyamatát átfogja. Az a rendszer, amelyet javasoltam a geometriai rajzok elkészítésétol kezdve a gyártásig támogatja a robotos ívhegesztést és mind a hegesztés-tervezonek, mind a robotos berendezés kezelojének folyamatos döntéstámogatást nyújt.

Az alábbi felsorolás mutatja be azokat az intelligens bovítéseket, amelyek kisebb (tervezési) vagy nagyobb (sikeres implementáció) mértékben a PROARC-hoz készítettem el [10]:

- A PROARC rendszerbe telepített technológia adatbázis képes arra, hogy default hegesztési értékeket, beállításokat ajánljon az operátornak. Ennek az adatbázisnak tudásbázisú kiterjesztése teszi lehetové a hegesztési "ökölszabályok" könnyu kezelését. A munkadarab geometriai alapadataitól és esetleg anyagjellemzoitol függoen javasolja az intelligens rendszer az egész varratra ill. az elemi varratdarabokra vonatkozó hegesztési beállításokat.

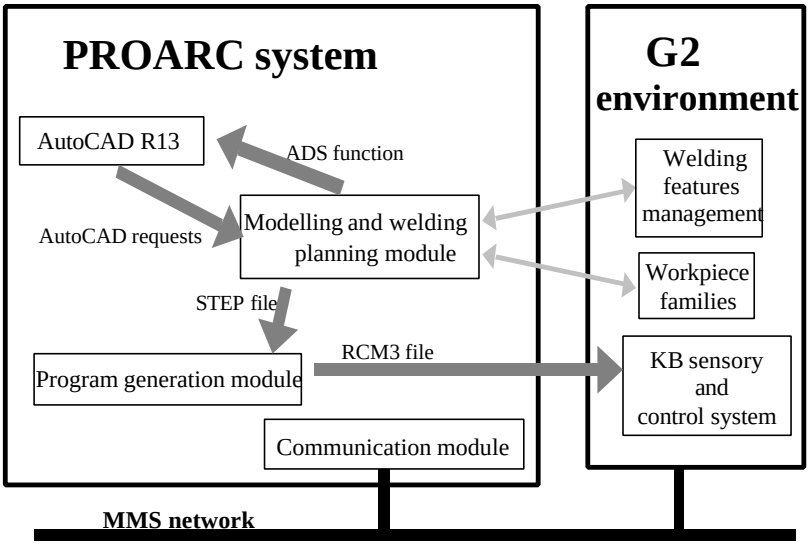
- A munkadarabok hasonlósága, a hasonlóság mértékének a meghatározása tipikusan az a feladat, amelyhez egy tudásbázisú rendszer támogatása nagy segítséget ad. Ha két munkadarab "hasonlónak" minosül, akkor az egyikben már beállított (kikísérletezett) hegesztési beállítások jó eséllyel a másik esetében is helyesek.

- A PROARC kommunikációs modulját tudtam a legegyszerubben kicserélni, hiszen erre kész volt az elvi megoldásom. A robotprogramot a G2-ben elkészített új cellavezérlo tölti le a robotba és vezérli távolról a robotot és a hegeszto-berendezést, indítja a hegeszto program futtatását, stb. A legenerált hegesztési robotprogramot eloször valódi hegesztés nélkül futtatja a rendszer, hogy a

kezelo ellenorizni tudja - a korábbi szimuláción túl - a robot által bejárt útvonalat.

- A hegeszto szenzorok adatainak intelligens feldolgozásával a hegesztés minoségét próbáltam javítani. A PROARC rendszer demonstrációs verziói mindössze tapintószenzorral rendelkeztek, de a szoftver további (pl. ív-) szenzorokat is támogatott. A tudásbázisú rendszerben a szenzor jelek intelligens feldolgozása nem elsosorban az on-line beavatkozást támogatja, sokkal inkább a szenzor jeleinek utólagos kiértékelését a hegesztés sikeressége, minosége ismeretében. Így egyfajta visszacsatolást biztosít a technológiai tudásbázis számára. Részletesen nem dolgoztam ki, de felvetem, hogy a varratok utólagos ellenorzése kiegészítheti az on-line minoségbiztosítást.

A legtöbb bovítésnek a terveit, specifikációit és kísérleti tudásbázisait készítettem el, mert a szükséges adatkonverzió munkaigénye, és az a szoftver feladat, hogy az AutoCAD-bol közvetlenül adatokat lehessen a G2-be eljuttatni, túlnott a prototípus keretein. A "proposal"-ben kidolgozott megoldásom itt is a STEP használatát helyezte elotérbe.



22. ábra A PROARC rendszer bovítése tudásbázisú rendszerrel

Az elkészült prototípussal igazoltam, hogy lehetséges és érdemes olyan tudásbázisú rendszert fejleszteni, amely az ívhegesztés teljes életciklusát támogatja. Egyetlen komplex, moduláris tudásbázisban lehet kezelni az összegyujtött tudást és adatokat egy olyan gépészeti területen, amelynek problémái technológiai szempontból nehezek. Gyakran ugyanis a szakértok is ellentmondásosan értékelik ki ugyanannak a technológiai problémának az adatait. A 22. ábra vékony szürke nyilai jelzik azokat a kapcsolatokat, amelyek csak terv és specifikáció szinten léteztek. A folytatás, amely a G2-nél egy lényegesen olcsóbb intelligens környezetet célzott meg, egy "intelligens" PROARC projekt azonban különbözo - nem muszaki - okokból nem valósult meg.

6.2 NYÍLT SZERSZÁMGÉP VEZÉRLÉS INTELLIGENS BOVÍTÉSE

A különféle nyílt gépvezérlések (4.2 alfejezet) programozói felületei - legalábbis az iniciatívát megfogalmazók szerint - biztosítják, hogy más gyártó megfelelo szoftverkomponense hozzákapcsolható a rendszerhez, a referencia modell pedig megadja, hogy mit kell az egyedi hozzáépített modulnak tudnia, hogy együtt is tudjon muködni a többi nyílt modullal. Így alkalmasak lehetnek MI alapú eszközök integrálásával intelligens CNC funkciók egyszeru kialakítására.

6.2.1 OSACA VEZÉRLO G2 ALAPÚ INTELLIGENS FELHASZNÁLÓI FELÜLETTEL

Az OSACA projekt második fázisában több kiegészíto modult fejlesztettek ki a résztvevok (pl. szerszámmenedzser, diagnosztika). Nem volt azonban addig olyan alkalmazás, ahol valamilyen komplexebb rendszert kapcsoltak volna az OSACA platformhoz. Az IDAS-OSACA (a projekt harmadik fázisa) keretében felvetettem, hogy az OSACA platform felhasználható intelligens CNC prototípusának kifejlesztésére is. A konzorcium támogatása mellett végeztem el ezt a munkát [22, 4] azzal a céllal, hogy bizonyítsam az OSACA koncepció életképességét az MI-t alkalmazó CNC-k világában is.

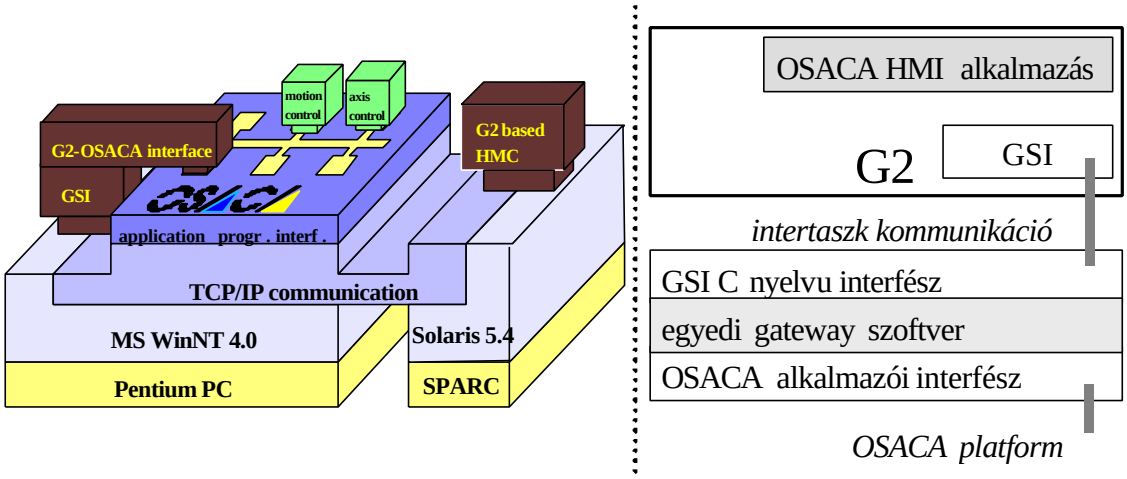
A kiindulást az OSACA egy Aachenben kifejlesztett egyszeru oktatási verziója jelentette, amely MS WinNT környezetben szimulálta egy CNC muködését (1 tengelyvezérlo (AC), 1 mozgásvezérlo (MC), 1 ember-gép kapcsolat modul (HMC)). Ebben a környezetben az ember-gép kapcsolat modult cseréltem le egy intelligens (G2 alapú) modulra.

A G2 alapú HMC modul a demó HMC minden funkcióját átvette. A legelso verzióm az adott pontokba való lineáris interpolációt és az elotolás értékének változtatását támogatta. A másodikban, amely az elozo SzTAKI-beli továbbfejlesztésén (Drozdik Szilveszter [51]) alapult, és 100 %-osan megfelelt az OSACA referencia architektúrának, néhány egyszerubb utasításból (G00, G01, X, Y, Z és F kód) álló NC programot lehetett végrehajtatni az MC, AC, HMC és MCM (mozgásvezérlo-szervezo) modulokkal.

Az egyszeru csatlakozást az a gateway szoftver tette lehetové, amellyel az OSACA alkalmazói interfészt a G2 rendszer interfészével (GSI) egy taszkban összekapcsoltam. Kihasználva az OSACA API ([1] 2. és 3. fejezete) felépítését, teljesen általános megoldást találtam, azaz bármilyen OSACA AO-t a GSI interfészen keresztül a G2-hez tudtam kapcsolni.

Az OSACA alkalmazói interfész ugyanis háromféle objektumosztályt ismer: változók, események és eljárások. A másik oldalon a GSI interfész a megoldások széles körét támogatja (külso változók írása és olvasása; külso eljárások hívása G2-bol; belso G2 eljárások hívása külso eljárásokból). Így mindhárom OSACA objektumosztályban, ha az adott muvelet kliens oldali, akkor a G2 a GSI külso eljáráson át éri el az OSACA API-t, míg a szerver oldalon a GSI-bol hívódik meg G2-ben levo eljárás, amely lekezeli a hívást. Nyilvánvalóan az intelligens rendszer lassabb feldolgozási sebessége határozza meg, hogy milyen modul realizálható benne; csak „kelloen lassú” ill. kliens jellegu modulok esetében van értelme a modult G2-ben implementálni.

A 23. ábra azt a korai struktúrát mutatja, amikor a G2 egy Sun SPARCstation-on futott, a második verzióban a G2 is átkerült a PC-re, és az egész rendszer egyetlen MS WinNT-n futott.



23. ábra G2 alapú HMC OSACA vezérloben

Ezek után a konkrét G2-es demóban, vagyis a HMC-ben kellett a többi OSACA modul (AC, MC, MMC) azon külso elérésu (*public*) változóinak, eseményeinek és metódusainak leképezéseit definiálni, amelyeket a HMC modul el akart érni. Ezek az objektumok azonnal elérhetové váltak a G2 tudásbázis számára. A következo G2-beli szabály is ezt illusztrálja:
whenever any axis AX receives a value and when the gsi-interface-status of g2-osaca-interface = 2 then start set_axis (the string of AX, the value of AX).

A G2-ben kialakított HMC prototípusban [23] az alábbi intelligens tulajdonságokat valósítottam meg:

- a tengely aktuális pozícióját és a sebességét egyszerre figyelembe vevo szabály alapú felügyelet
- tengely mozgásirányának fordulásakor maximum sebesség limit figyelés
- szimulált mozgási értékek adaptív vezérlésének szabály alapú hangolása.

Íme egy egyszeru szabály, amely adott tengelypozícióban a sebességet limitálja: *if the value of x_axis >= 3500 and the value of override >= 100 conclude that the value of override = 100*

Azóta, hogy publikáltam az elso intelligens OSACA modult [22], más feladatokban is választották ezt a platformot MI alapú CNC kutatási témákban. Ez történt pl. az IMS SIMON projektjében [158], ahol intelligens szerszámfelügyelet céljaira az OSACA referencia architektúrán túl további három modult (adatgyujto, feldolgozó és beavatkozó) specifikáltak.

7 AZ INTELLIGENS CNC

Az "intelligens CNC" kifejezés meglehetosen ritkán fordul elo a szakirodalomban. Ez nagyon meglepo figyelembe véve a CNC-k meghatározó szerepét a gyártásban. Sajnos a 10-20 évvel ezelotti elképzelések, remények nem realizálódtak a tényleges ipari hasznosítás területén. Nagyon sokat fejlodött a CNC és robottechnika, de az intelligencia oldalán nem sok valódi elorelépés történt. Az természetesen továbbra is igaz, hogy az intelligens gyártás egyik kulcsténye zoje lenne az intelligens CNC és robotvezérlés, ez a bevezetoben felsorolt IMS projekteket végignézve is nyilvánvaló.

Ha a nyílt vezérlés elnevezés használatával gondok vannak az irodalomban, akkor ez fokozottan igaz az intelligens CNC-re. Sok szerzo intelligens CNC-rol beszél akkor is, ha a vezérloben bármilyen formában MI algoritmus is beépítésre kerül egy hagyományos funkcióban (pl. fuzzy szabályozás egy PID algoritmus helyett). Mások akkor nevezik rendszerüket intelligensnek, ha a szokásos CNC funkciókon túl további kiegészíto funkció, bovítés is helyet kap a rendszerben, amely a szerszámgép muködését bármilyen szempontból biztonságosabbá, hatékonyabbá teszi, akkor is, ha ez a kiegészítés nem MI alapú.

Megtéveszto intelligens CNC editort emlegetni (pl. Predator [142]), ha az eszköz mindössze az NC program szerkesztéséhez ad olyan támogatást a CNC kezelojének, amelyet eddig a számítógépes szövegszerkesztok nyújtottak, de semmi több.

Hasonló szemlélet a vezérlésgyártóknál is megfigyelheto. Így például a legnagyobb gyártó, a GE Fanuc a rendszer intelligenciájának kialakítását továbbhárítja, mondván, ha azok az alkalmazói programok, amelyeket a CNC-ikbe (Pentium PC, MS Windows) lehet integrálni megfeleloen korszeruek, akkor ezek segítségével a szerszámgép intelligens gépnek tekintheto [63].

Ebben a fejezetben részletesen megvizsgálom az intelligens CNC kérdéskörét; az elvárásokat, a kutatási irányokat, majd a korlátokat és lehetoségeket. Közben értelmezem az ún. strukturális adaptivitás szintjeit a CNC-ben.

7.1 ELVÁRÁSOK

A felhasználó felol közelítve meg az intelligens CNC kérdéskörét, nem az a lényeges, hogy rejtetten milyen algoritmusokat használ az adott berendezés, hanem az, hogy mennyire rendelkezik olyan "képességekkel", amiktol valóban "intelligensnek" lehet nevezni. Természetesen nem az értekezés bevezetojében említett irreális elvárásokról van itt szó, amelyek szerint a vezérlés "mindent megold". Fontos kérdés az eladhatóság is, ami limitálja az eroforrásokat, hiszen kérdéses, hogy a felhasználó mennyivel ad(na) több pénzt egy intelligens CNC-ért a hagyományoshoz képest.

Hatvany József már 1983-ban felvetette [74], hogy MI alkalmazások segítségével lehetne a gyártórendszerekben megoldani váratlan, elore nem, ill. nehezen látható problémákat a rendelkezésre álló hiányos és nem teljesen pontos információ mellett.

Meg lehet állapítani, hogy közel 20 évvel késobb is lényegében élo kihívás Hatvany professzor javaslata.

Több, mint 10 éve gyujtötte össze Wright és Bourne [176] az intelligens megmunkáló géppel kapcsolatban az ipari elvárásokat. Az alábbi táblázatban jelzem azt is, hogy mely kívánalmakban történt jelentos fejlodés mára, valamint megmutatom, hogy hol jelentene az MI módszerek sikeres bevezetése nagy elorelépést.

Az intelligens CNC 1988-ban prognosztizált hatásai, tulajdonságai	jelentos fejlodés 2001-ig	MI módszerek szükségesek
csökkenti a kezdeti beállítást követo selejtes munkadarabok számát		✓
növeli a megmunkálás pontosságát	✓✓	✓
növeli a szerszámgép muködésének a kiszámíthatóságát	✓	✓
csökkenti azoknak a muveleteknek a számát, ahol emberi beavatkozás szükséges		✓✓
csökkenti a szerszámgép muködtetéséhez szükséges szakképzettséget		✓✓
csökkenti a munkadarab megmunkálásának összköltségét	✓	✓
csökkenti a szerszámgép állásidejét	✓	
növeli a gép teljesítményét (fajlagos ido alatt több megmunkált alkatrész)	✓✓	✓
többféle típusú munkadarabot képes beállítani és megmunkálni	✓	
többféle geometriájú munkadarabot képes megmunkálni		✓
jobb muveleti sorrendtervekkel csökkenti a felszerszámozás idejét, számát stb.	✓	
csökkenti a beállítás muveleteinek a számát,	✓	
csökkenti a beállítás idejét az ezt is figyelembevevo gyártmánytervezéssel	✓✓	
lecsökkenti a munkadarab tervezése és megmunkálása közötti idot	✓✓	
megnöveli a munkadarab tervezése és a vezérlés közötti információcsere mennyiségét és minoségét	✓✓	
megnöveli a megmunkálási folyamat és a vezérlés közötti információcsere mennyiségét és minoségét		✓
megnöveli a kezelo és a szerszámgép közötti információcsere mennyiségét és minoségét	✓✓	✓

15. táblázat Az intelligens megmunkáló gép jellemzoi Wright és Bourne szerint

Kritikusan értékelve a fenti listát látható, hogy az eredeti kritériumokban több pont nem is feltétlenül igényel intelligens viselkedést, és az elmúlt években sok területen (pl. pontosság; teljesítmény) nagymértékben javultak a szerszámgépek mutatói úgy, hogy a mesterséges intelligencia alkalmazásának a területén jelentos elorelépés nem történt. A fejlodés ugyanis két dolog miatt volt lehetséges: (1) a hardver fejlodése (nagyobb felbontású érzékelok, beavatkozók; nagyobb terhelések gyorsabb, pontosabb irányítása stb.) és (2) a szoftver fejlodése elsosorban a megmunkálást elokészíto tervezo (CAD, sorrend, ütemterv) környezetben, valamint a kényelmesebb és informatívabb

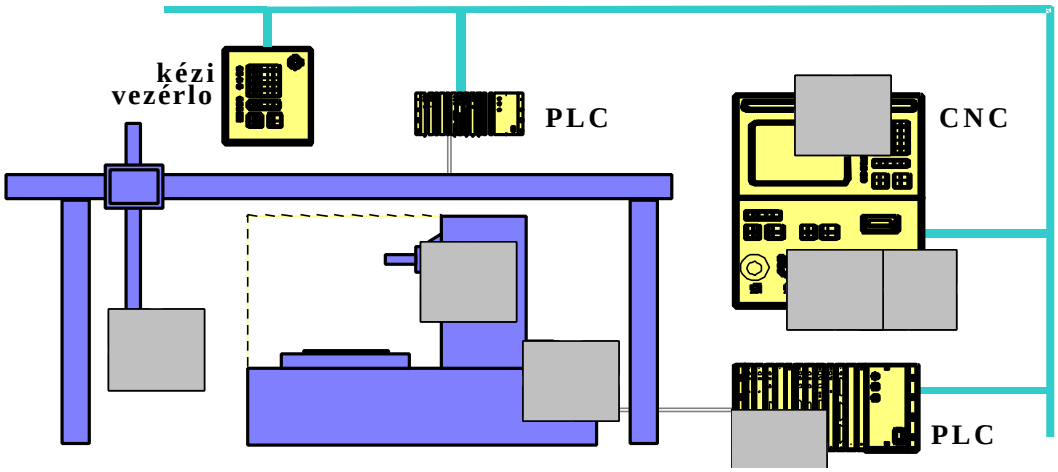
"Windows-szeru" felhasználói felület technológia a megjelenítokön. Ugyanakkor szinte mindegyik pontban megfogalmazott kritérium javítható lenne, ha a CNC "intelligensebb" lenne.

További fontos felhasználói szempontokat gyujtöttem össze [22] a következokben, ahol szintén látható, hogy az MI módszerek alkalmazása mennyire fontos lenne:

Egy intelligens CNC további tulajdonságai	MI segítene
modell alapú pályagenerálás	(✓)
automatikus szerszámválasztás	✓
megmunkálási paraméterek beállításának technológiai alapú automatikus támogatása	(✓)
a gép különbözo korlátainak automatikus figyelembevétele (sebesség, gyorsulás, terhelhetoség stb.)	(✓)
automatikus visszalépési stratégiák, mozgásvezérlés (félbehagyott megmunkálás folytatásakor, újrainduláskor, hiba esetén stb.),	✓
valamilyen geometriai eltérés (pl. befogás, munkadarab mérete) felismerése és a megváltozott szituációhoz való alkalmazkodás	✓✓
a vezérlés algoritmusainak on-line kiválasztása, paraméterezése, az adott megmunkálási feladathoz illesztése	(✓)
a muhely/gyár más elemeivel való intelligens (közös problémamegoldó) kooperáció	✓
a szerszámhiba (kopás, törés, hiány) felismerése után korrekció	✓✓
selejt automatikus kezelése (pl. újra-megmunkálással javítható, más munkadarabnak átminosítheto)	✓✓
a szerszámgép hatékony muködésének menedzselése, meghibásodás közeli állapotainak felismerése és korrigálása	✓✓
komplex öndiagnosztika (hardver, szoftver, megmunkálás)	✓✓

16. táblázat Egy intelligens CNC-tol elvárható további követelmények

A sor folytatható lenne a tanulási képességek figyelembevételével, de ez a kritérium rendszer önmagában is egy olyan intelligens vezérlot takar, amely "felismeri, hogy mi a gond", és "értelmesen és hatékonyan" megoldja úgy, hogy lehetoleg a környezete számára ez minimális változást, zavart jelentsen.



24. ábra Intelligens funkciók/szenzorok egy szerszámgép infrastruktúrában

A foorsón, a gép alapján, egyedi méropontokon lehet olyan szenzorokat elhelyezni, amelyeket CNC és/vagy a szorosan hozzáintegrált PLC valamely intelligens modulja dolgoz fel. Ugyanakkor a felhasználói felület (HMI) vagy a CNC valamely belso modulja is kaphat MI alapú támogatást (24. ábra).

Természetesen a különbözo jellemzok gyakran ellentétes kívánalmakat jelentenek, amelyek egymással konfliktusba kerülhetnek. Egy intelligens CNC-ben ezek kezelése elkerülhetetlen. Horváth Mátyás, Márkus András és Váncza József dolgozott ki eset alapú következtetésen alapuló rendszert, amely multiágens környezetben önti formába és kezeli le a konfliktusokat [78]. Ehhez hasonló megoldást lehetne itt is alkalmazni.

7.2 KUTATÁSI IRÁNYOK, EREDMÉNYEK

Sokféle egyedi kutatás folyik egymással párhuzamosan, amelyek célja az, hogy a CNC vezérlok egyes funkcióinak hatékonyságát valamilyen formában MI megoldással javítsák, például az alábbi területeken (felhasználva a [168]-t):

- Több megmunkálási paraméter (elotolás, fotengely fordulatszáma, fogásmélység) egyideju fuzzy logikán alapuló vezérlése [103] úgy, hogy nem szükséges más dinamikus modell kifejlesztése új gép, szerszám vagy munkadarab esetében.

- A CNC szabályozási algoritmusában neurális háló vagy fuzzy szabályok alkalmazása.

- Optimális pályatervezés, esetleg a pálya real-time korrigálása.

- A CNC saját hatékony kihasználtságát figyelve beleszól, hogy milyen megmunkálási feladatot kapjon (intelligens ágensek, ütemezés).

- Élettartam menedzselés, becslés szabály alapú rendszerrel, a CNC saját állapotának diagnosztizálása.

- Homérsékleti és egyéb deformációk kompenzálása neurális háló vagy fuzzy szabályok segítségével.

- Szerszámtörés detektálás (esetleg elorejelzés), szerszámkopás figyelés (esetleg kompenzálás), ehhez szakérto rendszer, neurális háló, fuzzy logika vagy intelligens szenzorok használata.

Nem MI algoritmusokkal, hanem szimulációval növeli meg a CNC "intelligenciáját" a TRUE-CNC [177]. A real-time szerszámgép-szimuláció a munkadarab állandóan változó geometriáját számolja, ugyanakkor folyamatosan számolja a fordulatonkénti anyagleválasztást. Ez utóbbit összehasonlítva az engedélyezett mértékkel és összevetve az egyes tengelyek egyedi korlátaival, a mozgásvezérlo az elérheto optimum közelében tudja tartani az elotolás értékét. Ugyancsak a TRUE-CNC része az ún. folyamat közbeni Gyors Dinamikus Kalibráció funkció (IQDC), amellyel az egyes tengelyek linearitását ellenorzi folyamatosan, és visszacsatolva az eredményt a mozgásvezérlobe a CNC kompenzálni tudja tengelyek eltérülését.

Egy összefoglaló cikkben Monostori László [118] három csoportba osztja a szerszámgépben lehetséges intelligens alkotóelemeket, és ismertet kutatási eredményeket: (1) szerszámállapot felügyelet (pl. fuzzy alapú mintafelismerés, osztályozás neurális hálókkal, káosz elméleten alapuló modell); (2) a megmunkálás és/vagy a megmunkáló gép modellezése (pl. neurális háló); 3) adaptív kontroll (hibrid MI, fuzzy).

Sajnos mindhárom területre igaz, hogy a kutatások és azok eredményei korlátozott megoldásokat biztosítanak. Ez sok esetben azt jelenti, hogy csak adott vagy egy viszonylag szuk tartományban levo gépre, munkadarabra, anyagra, beállításokra, szerszámokra stb. adnak értékelheto eredményt. Az általánosabb megoldások esetében is a jelentosen eltéro megmunkálási körülmények esetén újra el kell(ene) végezni azokat a kísérleteket, méréseket, amelyeket az eredeti vizsgálatokkor elvégeztek, hogy a másik tartományban is értékelheto, és a valós megmunkálás során (on-line) használható algoritmus álljon rendelkezésre. Érdemes észrevenni, hogy megoldást jelenthet, ha muködés közben lehet valamely algoritmust és/vagy paramétert lecserélni, újrakonfigurálni. Az ilyen igény a nyíltság témakörét helyezi elotérbe.

Több nyíltnak mondott kísérleti CNC architektúra, amelyek PC-be rakott külön DSP kártyát használnak a real-time feladatokra, rendelkezik intelligens felügyeleti kiegészítéssel (pl. adaptív forgácsolási ero szabályozás [35], vagy ugyanez kiegészítve homérsékleti deformáció kompenzációval [116]).

Ismeretesek olyan kísérleti "nyílt" CNC-k, ahol több intelligens funkció együttesen épül a rendszerbe. Cheng és társai szerint [44] az intelligens CNC abban különbözik a hagyományostól, hogy nagyobb a gép hatékonysága ill. jobb a minosége, valamint szerszámvédelmi rendszer is található benne. Rendszerüket nyíltnak nevezik, mert PC alapú, ISA buszos kártyákból áll, és az alaplapon MS Windows98-as operációs rendszer felett futnak rajta az MS Visual C++ alkalmazások. A PC alapú CNC rendszerük az alábbi kiegészíto - hardver - modulokat tartalmazza, hogy a szükséges real-time feltételeket teljesíteni tudja:

- Intelligens adaptív vezérlo modul, mely a forgácsolási erot optimalizálja a pillanatnyi ero értékek alapján fuzzy logika felhasználásával. [106]
- Tudásbázisú öndiagnosztika és hibaelhárító modul, amely a folyamat felügyelo szenzorok jelei és a jellemzo (feature) alapú folyamatleírás alapján következtet.
- Multiszenzor alapú szerszámfelügyelet hibrid (neurális háló és fuzzy) MI technológia segítségével.
- Mechanikai beavatkozóhoz tartozó hibakompenzáló, melynek elvérol nincs adat.

CNC holonikus gyártórendszerbe történo integrálásához a meglevo Philips vezérléshez kapcsolnak külön számítógépet, hogy az abban futó alkalmazás segítségével résztvehessen a CNC a holonikus gyártórendszerre jellemzo kommunikációban [166].

7.3 A STRUKTURÁLIS ADAPTIVITÁS SZINTJEI

Az adaptivitás értelemszeruen az intelligens viselkedés egyik elfogadott fokméroje. Az adaptív kontroll (AC) (pl. Horváth Mátyás és Somló János összefoglaló munkája [79]) a szerszámgépekben is régóta az egyik fontos MI alkalmazási terület, pl. a

SzTAKI-ban neurális hálóból és szakérto rendszerbol álló kísérleti rendszert [36] fejlesztettek ki.

Az elosztott rendszerek terén az informatika legújabb eredményei olyan új változásokat ill. lehetoségeket teremtettek, amelyek nincsenek kelloképpen kiaknázva gyártórendszerek körében. A Web technológia alapjai, a Java, a Corba, DCOM stb. a programok, modulok betöltéséról, inicializálásáról, futtatásáról meglévo fogalmakat gyökeresen átrendezték. Egy e-mail-ben vagy egy böngészoben egy link-re kattintva mindenféle feladatok (pl. Java applettek - programocskák) indulnak el, amelyeket éppen most tölt át a rendszer, vagy a szerveren, esetleg egy harmadik számítógépen futnak. Ez egyrészt a felhasználókat sokkal óvatosabbá teszi (pl. a megnövekedett vírusveszély miatt), másrészt látni kell, hogy az egész szoftver világ ilyen dinamikus muködési környezetbe került, amely megnövelt lehetoségeket (és természetesen problémákat is) teremt.

Felmerült bennem a kérdés, hogy ez az újfajta szoftvertechnológia hogyan jelenik meg a gyártórendszerekben, konkrétan a vezérlésekben, továbbá, hogy kínál-e megoldást olyan problémákra, amelyek jelen vannak a vezérlésekben.

Hagyományosan egy CNC minél intelligensebb, annál bonyolultabb, hiszen egyre több funkciót kell egyetlen berendezésbe integrálni. Az alapfunkciókhoz (pályagenerálás, szabályozások stb.) szorosan nem kapcsolódó funkciók ugyanakkor nem kellenek folyamatosan, hiszen pl. egy bonyolult 3D-s felület megmunkálásakor nem érdemes az elotolást optimalizálni AC-vel, hiszen a mért jelek is (pl. ero) nagyon függenek a szerszám pillanatról pillanatra változó irányától, továbbá a felület minosége is korlátozza a megmunkálási sebességeket. Ilyenkor, pl. a homérsékleti deformációk kompenzálásának lehet komoly szerepe. Ugyanakkor, pl. egy anyagleválasztó lineáris marásnál az AC optimalizálás idot, vagyis költséget takarít meg. Az egész berendezés lesz költségtakarékos, ha az egyes funkciók dinamikusan töltodnek be a rendszerbe, és csak akkor, amikor valóban szükség van rájuk. Természetesen ezt kísérleti rendszerben eddig is meg lehetett tenni, de észre kell venni, hogy ma ennek a megvalósításához szoftvertechnológiai oldalon minden készen áll.

Ha ebben a felfogásban értelmezem az adaptivitást, ez egyfajta folyamatos belso átkonfigurálását jelenti a vezérlonek, ahol adott modulok hol futnak, hol pedig nem. Lehetséges, hogy ugyanazt a funkciót dinamikusan váltakozva más - más szoftver taszk látja el. Megoldható az a probléma, hogy az egyes optimalizáló vagy intelligens algoritmusok csak korlátozott megmunkálási viszonyok között hatékonyak vagy muködoképesek. Ha egy adott kifinomult eljárás muködése lehetséges, akkor lecserélheto egy egyszerubb algoritmus erre, majd a viszonyok változása esetén vissza lehet az eredetihez térni, amikor esetleg a bonyolultabb, intelligensebb rosszabb eredményt jelentene.

A fentiekbol következoen a belso adaptivitásra az alábbi három szintet tudom értelmezni:

1. Nincs belso adaptivitása a berendezésnek, ha a saját struktúráját, algoritmusait nem képes valamilyen külso vagy belso hatásra megváltoztatni. A jelenlegi ipari gyakorlatban szinte minden berendezés ilyen.

2. Korlátozott a belso adaptivitása, ha a saját muködési paramétereit tudja változtatni, adott események hatására indít el / állít le funkciókat, adott funkciók prioritását változtatni tudja az igények szerint. Sok kísérleti rendszer jellemezheto ezzel a tulajdonsággal.

3. Teljesen adaptívnak tekintheto az a rendszer, mely akár moduljait is le tudja cserélni, újabb modulokat képes letölteni és elindítani, ha valamilyen változás erre készteti, vagyis lehetséges a berendezés dinamikus "testre szabhatósága".

Fontos megjegyezni, hogy egyes nyílt CNC rendszerek szoftveres megoldásai (4.2 fejezet) olyanok, hogy támogatják az itt bevezetett belso adaptivitást.

7.4 KORLÁTOK ÉS LEHETOSÉGEK

Miután bemutattam a kívánalmakat és a konkrét kutatási eredményeket, következik a kérdés, hogy mi az oka annak, hogy a jelenlegi CNC-k és robotvezérlok nem teljesítik a 10-20 évvel ezelott megfogalmazott és prognosztizált feltételeket? Ez annak ellenére igaz, hogy szinte mindegyik, a követelmények és elvárások között felsorolt funkcióval kapcsolatban található valamiféle kutatási eredmény. Az elozoek nyomán az alábbiakban tudom összefoglalni az észrevételeimet:

- Az MI módszerek nem képesek jelenleg átfogó, univerzális megoldást adni egy-egy problémára, ezért jelenleg általános, hogy minden környezetben valamiféle egyedi fejlesztésre van szükség, ami az ipari gyakorlatban megakadályozza az intelligens CNC-k kialakulását és elterjedését.

- Jelentos költséget jelent egy-egy berendezésbe integrálni a szükséges MI rendszereket, ami - szemben az elmúlt idovel - nem elsosorban a hardver (teljesítmény, memória, sebesség), hanem az integrálandó szoftver elemek költségeinek a kérdése.

- Nincs általánosan elfogadott vezérlés architektúra, amely az MI eszközök beültetését is egyszerusítené, bizonyos értelemben szabványosítaná. Bármennyire is sokan hangoztatják, nem kielégíto megoldás erre a kérdésre az, hogy a vezérlo hardver-szoftver platformja (infrastruktúrája) nyílt és könnyen hozzáférheto (pl. PC + MS Windows + SERCOS stb.).

- Nincs kelloen kiaknázva a diagnosztika szerepe. Jelenti ez a berendezés muködésével kapcsolatos elemzéseket (hardver, szoftver), és annak a vizsgálatát, hogy a valós megmunkálás mennyiben tér el a attól , amit a berendezés programjai és beállításai alapján eloírtak neki.

- A vezérlésekben futó szoftverek nem képesek dinamikusan alkalmazkodni a változó körülményekhez.

- Nincsenek általánosan elfogadott módszerek a felügyeleti funkciók kapcsán a szerszámgép "felszenzorozására" és a jelek kiértékelésére, ami az egyedi megoldások szélesköru használhatóságát gátolja. Eseti, hogy egy

megmunkálógépen vannak-e elérheto szenzorok, azok pontosan hol helyezkednek el, mit mérnek.

- A gép és cellavezérlok közötti intelligens kommunikáció megoldatlan. Nincs arra lehetoség, hogy más rendszerekben bekövetkezett devianciák "tapasztalatait" átvehesse a vezérlés. Hasonló kérdés, hogy pl. pontos információval rendelkezve saját szerszámainak állapotát illetoen beleszólhat-e abba, hogy milyen megmunkálásokat kapjon?

- A jelenleg elterjedten használt NC programnyelv (valamilyen G kód) nem ad a munkadarabról annyi információt, ami adott helyzetekben szükséges lenne intelligens döntések meghozatalára (selejt esetleges javítása, befogási hibák kezelése; megmunkálás felfüggesztése, újraindulása stb.). A vezérlések tipikusan nem használják a CAD rendszer által szolgáltatott munkadarab modellt.

A kérdés értelemszeruen az, hogy ezeket a korlátokat lehet-e új IT megoldásokkal feszegetni, enyhíteni, esetleg megoldani.

Az univerzális megoldások hiánya látható több korlát mögött, amire a szabványos architektúra jelent megoldást itt is, amint az más területeken már sokszor beigazolódott. Nyílt és szabványos architektúrákban kidolgozott egyedi jellegu megoldások sokkal egyszerubben vihetok át az egyik alkalmazásból a másikba. Ez azonban csak akkor igaz, ha nem csak az infrastruktúra nyílt, mint a jelenleg kapható PC - MS Windows vezérlésekben, hanem modul szinten is fennáll a nyíltság. Ez a fajta újrafelhasználhatóság fogja a költségeket is érdemben csökkenteni, ami elvileg még hatékonyabb eroforrás gazdálkodással is javítható.

Az adaptivitás elérése az infrastruktúra terén ma már nem igényel különleges informatikai környezetet. Ha a modul szintu nyílt architektúra ténylegesen ilyen környezetben muködik, akkor a 7.3 alfejezetben leírt harmadik szintu adaptivitáshoz szükséges háttér megvan, a megfelelo modulok segítségével az egész azonnal realizálható. Továbbá a nyílt infrastruktúrában realizált vezérlo rendelkezik azzal a tulajdonsággal, hogy az esetleges intelligens bovítések könnyedén - akár futási idoben is - hozzáépíthetok a szabványos elemekhez. Ez a fajta struktúra támogatja mindenféle diagnosztikai funkció vagy bármilyen eseményfüggo modul alkalomszeru betöltését, elindítását, majd a muködése után leállítását.

Érdemes azt is megvizsgálni, hogy az egyes MI módszerek milyen informatikai követelményeket támasztanak a (vezérlések) lehetséges infrastruktúrájával (hardver, operációs rendszer, egyedi szoftver környezet stb.) szemben. A vizsgálatok az on-line muködés során használható eljárásokat célozzák. Nyilvánvaló az is, hogy bizonyos MI módszerek (pl. genetikus algoritmusok) nem jöhetnek szóba egy CNC vagy robotvezérlo valamely intelligens komponensében, és ez jelentosen lecsökkenti a vizsgálandó módszerek körét. A hibrid rendszerek (2.1.3 alfejezet) számítástechnikai jellemzoi - értelemszeruen - a bennük használt "tiszta" módszerek ilyen jellemzoibol adódnak össze.

Továbbá az is igaz, hogy több MI módszer muködési logikája egy tanulási/preparálási és egy végrehajtási/muködési fázisra osztható, a számítási igény és körülmények figyelembevételénél azonban az on-line muködés miatt csak az utóbbi fázis követelményeivel kell számolni. Pl. egy neurális hálózat tanításához sokkal több

ido és számítási kapacitás szükséges, mint a már betanult hálózat muködtetéséhez (a kimeneteinek a meghatározásához) új jelek alapján.

A 17. táblázat csoportonként összefoglalja a különbözo MI eljárások számítástechnikai jellemzoit. A táblázat egyes bejegyzéseivel kapcsolatban természetesen fennáll egyfajta szórás, de általában a megadott jellemzok a tipikusak.

módszerek	ismeret alapú	szubszimbolikus	modell alapú
számítási igény	inkább nagy	(on-line) inkább kicsi	nagy
számítási ido	méretfüggo	gyors	méretfüggo
speciális szoftver komponens igény	problémamegoldó következtetogép, szabályok;	nem feltétlenül szükséges	modell leírás, modell építés, kezelés
feldolgozási sebesség	méretfüggo	közel real-time	lassú
adatmennyiség	közepes	közepes	közepes
feldolgozás módja	esemény orientált	esemény orientált / ciklikus	esemény orientált

17. táblázat Mesterséges intelligencia módszerek számítástechnikai jellemzoi

Látható, hogy a szubszimbolikus módszerek azok, amelyek általában nem igényelnek speciális szoftver elemeket, és amelyek alkalmasak hardver közeli valós ideju adatfeldolgozásra. Egyedi programozással kifejleszthetо vagy fejleszto rendszerek segítségével automatikusan legenerálható olyan szoftver komponens, amely adott feladatra valamilyen szubszimbolikus alapú megoldást valósítanak meg, és így ezeket egy gépvezérlés hardver közeli számításainál fel lehet használni.

Két csoportba oszthatók tehát az intelligens funkciók a realizálhatóság szempontjából. Az egyikbe olyan funkciók kerülnek, amelyeknél a konkrét - tipikusan szubszimbolikus - MI módszer on-line végrehajtása lehetséges egy black-box jellegu önálló szoftver komponensben, és ráadásul olyan eroforrásigények lépnek fel (sebesség, teljesítmény stb.), amelyek kielégíthetok egy hagyományos algoritmusú CNC infrastruktúrában, nem igényelnek extra eroforrásokat. A másik csoport funkciói esetén nem lehetséges egy viszonylag egyszeru black-box komponens feltételezése, mert mindenképpen szükség van valamilyen nagyobb szoftver eroforrásra (pl. problémamegoldó, modellezo, esetleg tanulást vezérlo stb.). Pontosabban még az is lehetséges, hogy a második csoport egyes elemeit fel kell bontani egy egyszerubb adatgyujto - elso csoportbeli - és egy maradék komplex - második csoportbeli - részre.

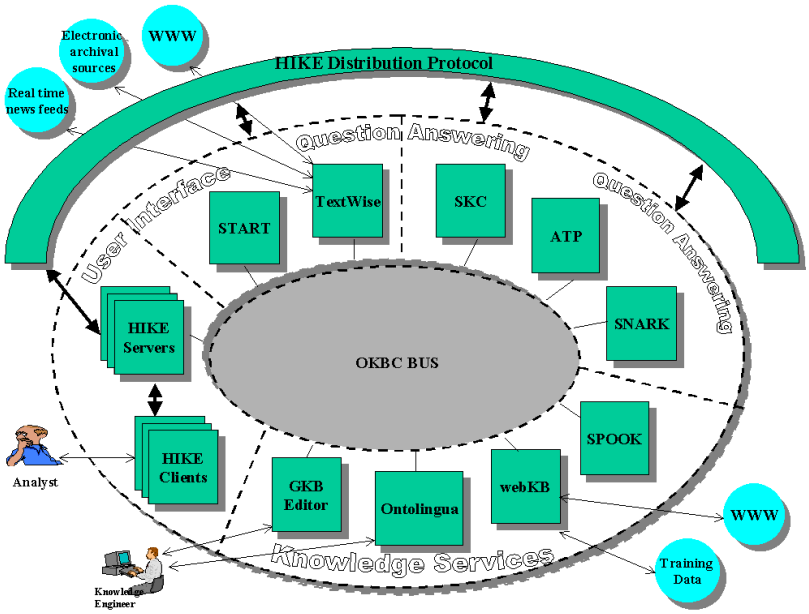
Mivel a második csoportban szereplo funkciók nem ido kritikusak és dominánsan eseményvezéreltek, ezért lehetséges olyan CNC infrastruktúra kialakítása, hogy ezek feldolgozása egy távoli - eroforrása miatt alkalmas - szerveren történjen, amelyik helyi hálózaton kapcsolódik a CNC-hez. A koncepció részletes kidolgozását mutatja be a 8. fejezet.

Nem állítom, hogy mindig, minden esetben magvalósítható ez a felosztás, véleményem szerint annyi is elegendo, hogy az esetek dönto részében megteheto, és ezzel kialakítható egy olyan CNC architektúra, amely alkalmas keretet jelent az intelligens CNC-k kialakítására.

8 VEZÉRLÉSEK TUDÁSSZERVERE GYÁRTÓRENDSZEREKBEN

A tudásbázisú rendszerek kialakulását az tette lehetové, hogy az absztrakt következteto gép és a szabályhalmaz egymástól különválasztható és így lehetoség van általános következteto gép algoritmusokat implementálni a felhasználóra bízva az alkalmazáshoz tartozó szabályok implementálását.

A World Wide Web újszeru lehetoségei vezették Erikssont [58] annak a felismerésére, hogy több, a szakérto rendszerekben meglévo problémák (installálási és verziófrissítési gondok, elosztott és távoli hozzáférések) hatékonyan orvosolhatók lennének, ha a szakérto rendszerek felhasználói felületét a Web technológia biztosítaná. Ezt a koncepcióját nevezte tudásszervernek. Ilyen elven muködik pl. az Istar [37] tudásszerver, amelyet a Salford Egyetem Információs Rendszerek Intézete üzemeltet, és on-line tanácsokat ad tozsdei (érdemes-e venni valamilyen részvényt vagy sem), Internetes üzenetek biztonságtechnikai és faültetési (hova milyen fákat érdemes ültetni) problémákban.



25. ábra HIKE architektúra a különféle tudásszerverekkel a HPKB projektben [104]

Az MI alkalmazások csúcsát jelento amerikai hadipari kutatások (DARPA) között szerepel a nagyteljesítményu tudásbázisokat épito HPKB projekt (High Performance Knowledge Environment), amelynek célja százezer - millió szabállyal muködo nagy tudásbázisok megalkotása. Itt a különbözo intelligens eszközöket nevezik tudásszervereknek, amelyek egy integrált tudáskörnyezetben (HIKE - HPKB Integrated Knowledge Environment) [104] muködnek együtt. A különbözo komponensek között a tudásmegosztás számára a Stanfordon specifikált OKBC-vel (Open Knowledge Base Connectivity) [43] vagy különféle hagyományos (http, Java RMI, socket) protokollokkal kommunikálnak.

Hasonlóan nagyteljesítményu környezetet jelent a mindennapi tudás (common sense) kutatásában vezeto Cyc rendszer [102], amely szintén használja a tudásszerver

fogalmát. A Cycben ez egy kifejezetten nagy, sokféle szövegösszefüggést leíró tudásbázist és következteto gépet jelent, amely köré kapcsolódik a CycL tudásreprezentációs nyelv, a természetes nyelv feldolgozására alkalmas alrendszer és a különféle jelentéstanokat integráló Semantic Integration Bus. A Cyc a HPKB projektben is szerepet kapott.

A gépészeti alkalmazásokhoz közeledve említheto az intelligens CAD rendszerek közül az autótervezésben ismert ICAD rendszerbol kifejlodött KBO (Knowledge Based Organisation) Environment [94], amelynek tudáskezelo moduljai között szintén található egy tudásszerver, ami a rendszer talán legfontosabb részeként a CAD adatok tudásalapú feldolgozását végzi.

Váncza József eset alapú következteto rendszere [173], amely robotos átvizsgálás tervezés optimalizálására készült, használ tudásszerver alapú architektúrát, ahol a felhasználók valamilyen böngészo program segítségével az Interneten keresztül érik el a rendszert.

Az eddigi rendszerek mind a Web technológia használatán alapulnak. Ezektol eltér az a környezet, amelyben korlátozott eroforrású számítógépek - amelyek tipikusan mobil környezetben muködnek (pl. palm-top-ok) és elsosorban adatgyujtésre szolgálnak - nem alkalmasak arra, hogy a jelenleg használatos döntéstámogató rendszereket futtassák. Ezért több olyan kutatási projekt van, ahol a hagyományos környezetben (nagyobb számítási kapacitással és eroforrásokkal rendelkezo számítógépen) futó MI alkalmazást valamilyen egyszerubb környezetben levo egységgel összekapcsolva biztosítják, hogy abban a környezetben is elérhetok legyenek az MI rendszer szolgáltatásai. Nem a hagyományos távoli terminál jellegu szolgáltatásokra kell itt gondolni, bár erre hosszú évek óta sok példát lehet mondani, hanem olyan rendszerekre, ahol a központban futó MI alkalmazás tudásszerverként funkcionálva növeli a helyi eroforrás intelligenciáját, és így segíti a felhasználó munkáját.

Ötletes alkalmazások sora ismert, pl. a Veridian Engineering balesetek helyszíni elemzéséhez fejlesztett kéziszámítógépet és hozzá egy szakérto rendszert [174] használ, amely távolról támogatja az intézkedo rendort, csökkenti a helyszíni kézi adatbevitelt, mégis pl. meg tudja állapítani, hogy a sofor biztonsági öve be volt-e kapcsolva.

Az Edinburgh-i Egyetemen kifejlesztett Java alapú rendszer [48] egy hagyományos döntéstámogató rendszert (ThiCon) kapcsol össze egy mobil tanácsadó rendszerrel (MoThA). A ThiCon rendszerben több tudásbázis modell és adathalmaz található, amelybol egyidoben egyet telepítenek a MoThA rendszerbe, amit az éppen beérkezo adathalmaz feldolgozásához használni kívánnak. A konkrét esetben erdogazdálkodásra használják a rendszert, és egy humán tervezo dönti el, hogy mikor melyik tudásbázist töltik be. Nyilván semmilyen elvi akadálya nincs más alkalmazásoknak ill. – az alkalmazástól függoen, hogy a tudásbázis kiválasztás se legyen emberhez kötött.

Komplex tudásbázis részleges telepítése egy csökkentett eroforrású környezetbe többféle módon lehetséges.

- A legkézenfekvobb, ha a csökkentett eroforrású számítógépben is fut egy következteto gép, és akkor a tudásbázis redukciója biztosítja, mely a felhasznált modellek egyszerusíto paraméterezésével lehetséges, hogy a kisebb környezetben is muködoképes maradjon a következtetés.

- Programkonverzióval megoldható, hogy a dekleratív tudásbázis egyszerusített (parametrizált) változatát procedurális kóddá konvertálja át a rendszer. A ThiCon-MoThA esetében a Prológ alapú dekleratív tudásbázist a telepítéskor Java alapú procedurális kóddá konvertálja át a rendszer.

- Lehetséges, ha a tudásbázis már önmagában tartalmaz olyan procedurális elemeket, amelyeket dinamikusan letöltve a target eszközre a komplex eszköz többszörös program letöltéssel, eredmény lekérdezéssel mintegy távolról "levezényel" egy következtetési folyamatot [64].

A mai elosztott programozási technológiák nagymértékben megkönnyítik az utóbbi két esetben vázolt technológiák alkalmazhatóságát.

8.1 A VEZÉRLÉSEK TUDÁSSZERVERE KONCEPCIÓ

A nyílt intelligens vezérlések korábban leírt prototípusainak (6. fejezet) az elemzése a gyakorlati alkalmazhatóság szempontjából több kérdést vet fel. Így ahhoz, hogy a vezérlés alapfunkciói mellett futtatni tudja az intelligens moduljait (pl. következteto gép stb.) a vezérlés hardverének sokkal nagyobb teljesítményunek kell lennie. Bár ennek is nyílván költség vonzata van, de a szoftver szempontjából még rosszabb a helyzet. Minden intelligens vezérlésbe szükség van (kifejezetten költséges) szoftver komponensekre, amelyek alkalmasak az intelligens algoritmusokat hatékonyan futtatni.

Mindezek alapján a tudásszerver koncepciót - az eddigiek analógiájára - bevezetem a vezérlések körében is. Definícióm szerint a **vezérlések tudásszervere**, a KSC (Knowledge Server for Controllers) egy megfelelo számítás kapacitással muködo szerveren futó intelligens környezet, amely alkalmas MI algoritmusok hatékony futtatására és így ilyen feladatokat képes átvállalni egy kapacitásában szukösebb vezérléstol. Adott esetben az is kapacitás korlátot jelent, ha egy adott szoftver nem fut rajta pl. licence okok miatt. Pontosan az MI futtatási környezet biztosítása a tudásszerver koncepció lényege.

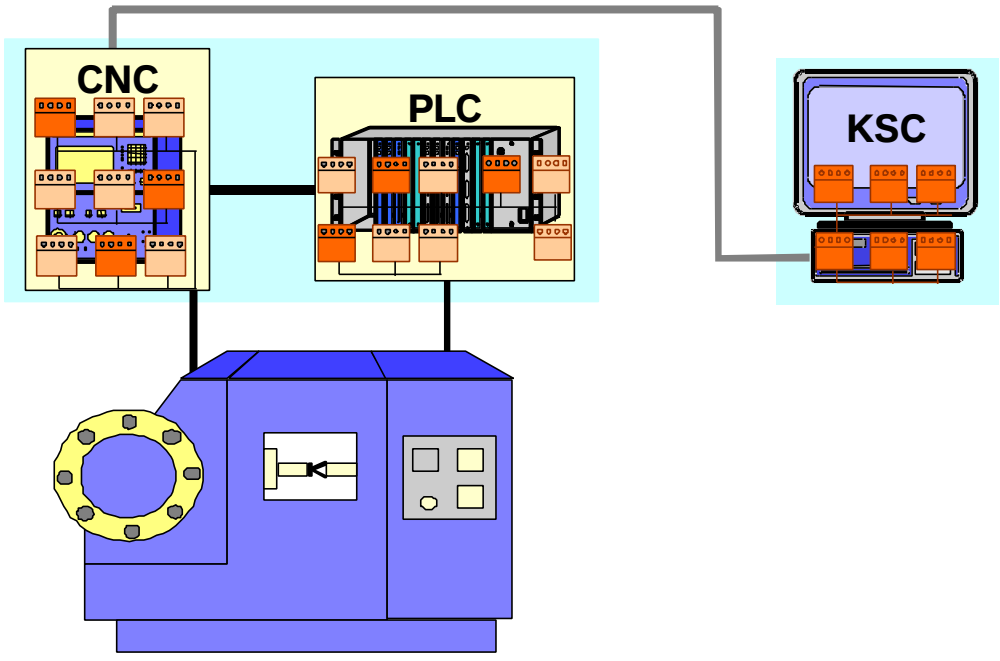
A KSC-ben több, önálló feladatot ellátó modul fut, amelyek megosztoznak a szerver MI eroforrásán. Implementációs szempontból ennek a követelménynek egy olyan MI keretrendszer tud a legegyszerubben megfelelni, amelyben egymástól független tudásmodulok találhatók. Minden egyes modul a saját feladatát végzi a meglévo tudása és a vezérlotol érkezo konkrét adatok alapján. Ez a felépítés lehetové teszi a modulok dinamikus menedzselését (7.3 alfejezet). Lehetséges, hogy egy modul, betöltve a szerverbe, valamilyen indító eseményre - a vezérléstol érkezo hívásra - vár. A mai keretrendszerek ugyanakkor akár arra is alkalmasak, hogy nagyobb modulok a hívás hatására töltodjenek be a rendszerbe, vagy - ha már nincs rá szükség, akkor - törlodjenek a rendszerbol.

Szemben a HPKS, Cyc stb. nagy rendszerekkel nálam - ez a KSC-nek az egyik legfobb újdonsága - a szerver kliensei nem humán felhasználók, hanem önálló számítógépek, pontosabban vezérlések (CNC-k, robot vezérlok, PLC-k stb.), ugyanakkor a tudásszervernek is lényegesen kisebb kapacitásúnak kell lennie, hiszen a célterület (intelligens vezérlés) is jelentosen kisebb és modulárisabb MI tudásbázisokat és algoritmusokat határoz meg. A példaképpen bemutatott korlátozott eroforrású, mobil

rendszerekkel összevetve pedig jelentos különbség, hogy a vezérlések esetében azért lényegesen nagyobb eroforrások állnak rendelkezésre, mint egy palm-top számítógép. A mobilitás is mást jelent, hiszen mobil robotok esetén is a kommunikációs kapcsolat a gyakorlatban a hagyományos LAN kapcsolatot biztosító rádiókapcsolat.

Itt mutatkozik meg - visszatérve 7.4. alfejezethez - a kétféle csoportba osztott MI eljárásoknál a csoportosításom értelme. Kihasználva egy nyílt architektúrájú CNC elonyeit (4. fejezet), a pontosan specifikált modul architektúra lehetové teszi, hogy a második csoport minden szükséges eleme egy külso, nagy teljesítményu és MI alkalmazások futtatására felkészített számítógépre kerüljön át (pl. egy G2 keretrendszer moduljaként). Az ilyen csoportosításnak megfelelo infrastruktúrát illusztrál a 26. ábra. Elképzelheto, hogy (1) egy intelligens modul teljes egészében a KSC-n fut, de az is, hogyha mélyebben a vezérlésbe tagozódik, akkor (2) egy része a vezérlésben (a nem MI rész) és csak az MI rész a KSC-n. Igazából ezt implementációs részletkérdésnek tekintem. Tudásbázis részleges telepítésére - a fejezet bevezetojében leírt második két módszer - is használható értelemszeruen a KSC struktúrát alkalmazó vezérlésekben.

Felhívom a figyelmet, hogy több CNC megosztozhat egyetlen KSC-n, ami azonnal - a megosztás mértékében - csökkenti az egy CNC-re jutó költségeket. Figyelembe véve, hogy intelligens CNC-k várhatóan nagyobb muhelyekben, gyártórendszerekben kerülnek leginkább alkalmazásra, reális a közös szerver koncepciója. A több CNC kiszolgálása miatt javaslom a KSC-be realizált MI funkciókat úgy implementálni, hogy egy konkrét feladatú funkció több vezérlést is ki tudjon szolgálni. Emiatt - bár ez most csak becsülheto - elotérbe kerülnek a (2) típusú modulok. A 8.4 alfejezet prototípusában is egy ilyen modult realizáltam.



26. ábra Nyílt architektúrájú intelligens CNC részben külso modulokkal

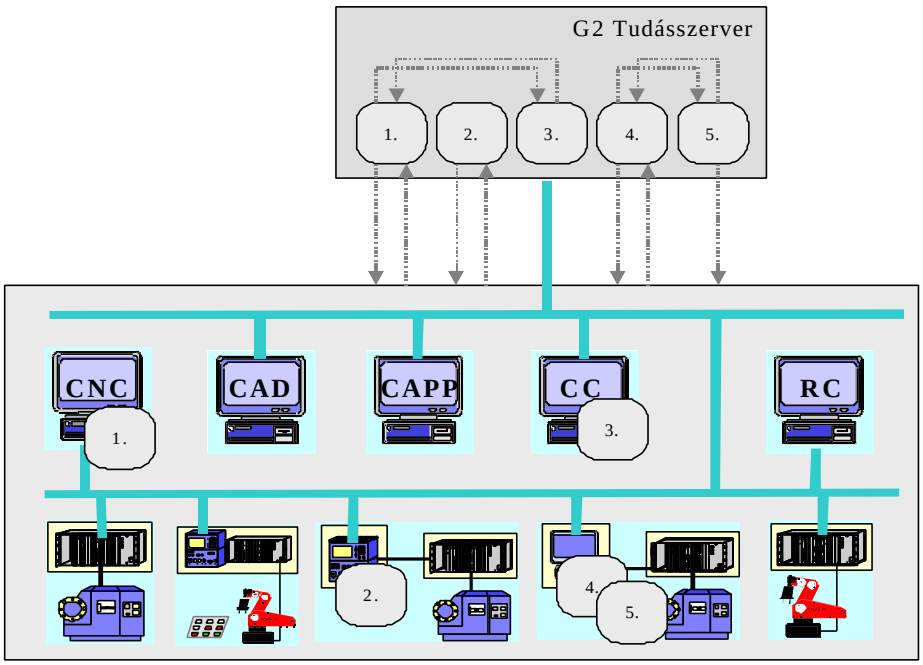
A CNC-ben, vagy a hozzátartozó PLC-ben futnak a hagyományos modulok (az ábrán világos dobozok), és az elso csoportbeli intelligens algoritmussal rendelkezok (az ábrán sötét dobozok), amelyek egy hagyományos funkciót helyettesíthetnek, vagy újabb

funkciókat valósíthatnak meg. A CNC további intelligens moduljai (második csoport) a helyi hálózattal összekapcsolt külso MI számítógépen, a KSC szerveren.

Más oldalról is megközelítettem a két csoportra osztást. Javaslom megvizsgálni, hogy a számításigény, a speciális szoftverek és a real-time követelmények miatt mik azok az intelligens funkciók (esetleg csak részfunkciók), amelyeket mindenképpen a CNC-ben kell tartani, és a többit "ki lehet hozni" egy megfeleloen eros számítógépre, amely fel van készítve a szükséges feladatok végrehajtására. A mai minimum 100 Mbit/s-os helyi hálózaton a két gép (CNC és KSC gép) közötti kommunikáció nem jelent problémát sem a sebesség, sem az adatfluxus szempontjából.

Szeretném kihangsúlyozni, hogy koncepcióm nem egy meglévo intelligens cellavezérlo és az egyszerubb CNC-k felállásnak az újrafogalmazása. Itt a külso gép nem vezérel, hanem eroforrásként az intelligens adatfeldolgozási képességét (következteto gép, modell alkotás, tanulási mechanizmusok) szolgáltatja a CNC berendezés számára.

Ha a KSC-hez kapcsolódó gépvezérlések ráadásul egy korszeru, komponens alapú szoftver infrastruktúrán futnak (pl. Corba, Java stb.), akkor az azt is lehetové teszi, hogy a black-box alapon felfogott modulokat futási idoben le lehessen cserélni. Meglátásom szerint ez abban a problémában segít, hogy sok konkrétan kidolgozott MI alapú eljárás csak adott korlátok között használható. Így viszont dinamikusan le lehet cserélni az algoritmusokat, ha a megmunkálási feltételek úgy változnak, hogy ott már az adott algoritmus nem érvényes.



27. ábra KSC gyártórendszer környezetben

A számítástechnikai környezet és a KSC kapcsolódása elvileg neuralgikus pontja lehet a koncepciónak. Itt is a nyíltságot látom megoldásnak, vagyis nem szabad valamilyen egyedi fejlesztésu interfésszel megoldani a kapcsolódást. Érdemes az OSACA projekt (4.2.1 alfejezet) hibájából - specifikus OSACA platform kapcsolja a

modulokat a külvilághoz - tanulni. A 27. ábra egy több berendezésbol álló gyártórendszerben mutatja a KSC (itt konkrétan G2 alapú) illeszkedését. Azt is leszögezem, hogy elvileg semmi akadálya nincsen annak, hogy a KSC CNC-n és RC-n kívül más - magasabb szintu - berendezéseket is kiszolgáljon (pl. CAD, CAPP, CC). Az ábrán arra is utalok, hogy az egyes intelligens modulok (pl. egy ágens elven muködo rendszerben) egymás számára is szolgáltathatnak adatokat.

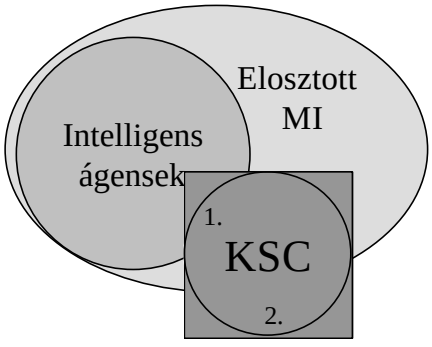
A KSC koncepcióm tehát választ ad az intelligens CNC-kkel kapcsolatos egyik legkomolyabb problémára, hogy milyen architektúrában lehet hatékonyan ilyen CNC-ket építeni. A 4. fejezetben bemutatott sokféle nyílt rendszer architektúra mutatja, hogy sajnos nagy a divergencia ebben a kérdésben, ezért a KSC koncepció bevezetésénél nem érdemes egy "százegyediket" bevezetni. Felhasználva a 4.2.4 alfejezet elemzését az OMAC alapú építkezést tartom a helyes választásnak. A részletes kidolgozás második lépése a különbözo MI funkciók interfész szintu specifikációja, hogy el lehessen készíteni egy kiindulási készletet, amellyel hatékonyan lehet ipari felhasználású CNC-ket építeni. Ezt a munkát elkezdtem (8.4 alfejezet) [33].

A KSC kapacitása alkalmas arra, hogy a jelenlegi szintnél sokkal hatékonyabb technológiai tudásbázis, modellezési technika és diagnosztikai eroforrás váljon elérhetové a CNC-k számára.

Diagnosztikai feladatokat kiválóan el tud látni egy KSC-be telepített modul, amely a vezérlések számára különféle hibaelhárító stratégiákat szolgáltat egy gyujtemény [164] segítségével.

8.2 A KSC ÉS AZ INTELLIGENS ÁGENSEK ÖSSZEHASONLÍTÁSA

Fel akarom hívni a figyelmet az elosztott és/vagy intelligens ágensek és a KSC közötti különbségekre és hasonlóságokra. Az ágensek önmagukban autonóm tulajdonsággal bíró entitások, amik közösen alkalmasak egy-egy feladat hatékony végrehajtására pont azáltal, hogy miközben lokális céljaikra törekednek, ennek eredményeképpen az egész rendszer számára is remélhetoleg korrekt és jó globális megoldás alakul ki. A KSC koncepcióban egyszeruen a tudáskezelés és az MI algoritmusok hatékony futtatásáról van szó egy erre kifejezetten alkalmas környezetben. Ebbol a szempontból alkalmazási kérdés, hogy a szervert használó programok, modulok stb. ágensekként viselkednek-e vagy sem.

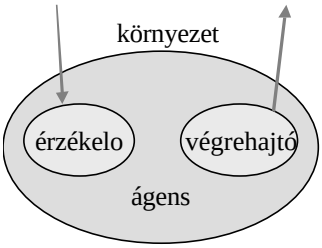
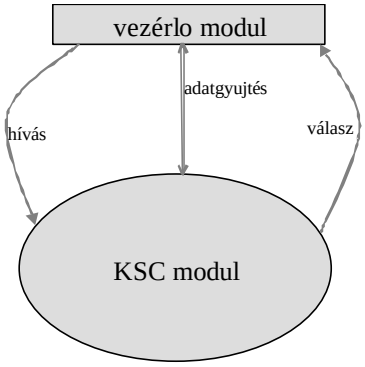


28. ábra KSC elhelyezése az elosztott MI világában

A 28. ábra mutatja, hogy a KSC alapú rendszerek alapvetoen az elosztott MI világába tartoznak, de lehetséges, hogy:

1. a KSC által támogatott rendszerek önmagukban lokális MI alkalmazások, így a valóságban nem elosztott az intelligencia, abban az értelemben, hogy kooperálnak az alkalmazások,
2. a KSC-t használó autonóm modulok intelligens ágensekként viselkednek, egymással kooperálnak, mesterséges személyiségük van, noha szoftvertechnológiailag ugyanazt az eroforrást használják.

A 18. táblázatban összehasonlítom a KSC és az ágenstechnológia fontos tulajdonságait:

intelligens ágensek	**KSC modul**
	 hasonlóságokat mutat az egyszeru reflex ágens [61] belso struktúrájával
érzékel	meghívják
autonóm	szerviz
kezdeményezhet	(alapértelmezésben)nem kezdeményez
célja van	feladata van
temporális kontinuitás	esemény éleszti fel, egyébként inaktív
opcionális tulajdonságok: mesterséges személyiség, kooperáció, tanulás, adaptivitás, stb.	opcionális tulajdonságok: ágens módon való viselkedés

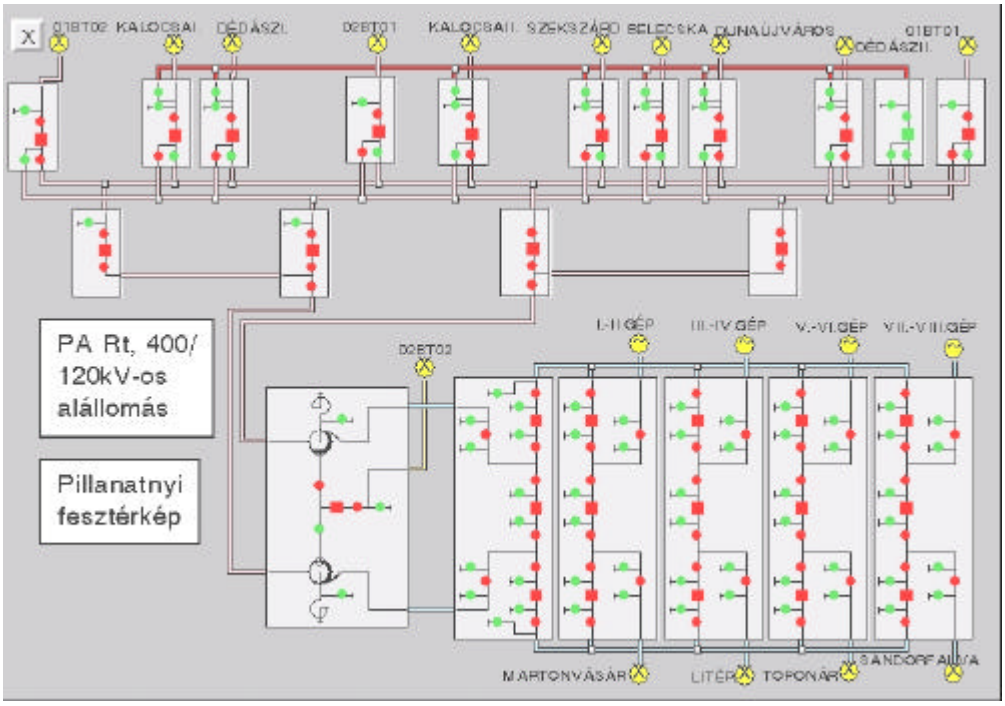
18. táblázat Az intelligens ágensek és a KSC tulajdonságai

Az ágens *céljaira* tekintettel *önmaga* dönti el, hogy az *érzékelt* események, üzenetek alapján milyen aktivitást *kezdeményez*. A KSC modul *szerviz*, vagyis konkrét *feladatot* képes végrehajtani, ha ezt valaki *meghívja*, magától semmit *nem kezdeményez* alapértelmezésben. Mivel a KSC egyfajta számítási mód, nincs elvi akadálya annak, hogy moduljait párba állítva az oket hívó nem MI alapú modulokkal, a kettot egy párban ágensként használjuk.

8.3 KONKRÉT PÉLDA - TANÁCSADÓ RENDSZER A PAKSI ATOMEROMUBEN

A Paksi Atomeromu villamos alállomásán a Magyar Villamosmuvek (MVM Rt.) kezdeményezésére és támogatásával az elmúlt években nagyfokú üzemeltetési rekonstrukció zajlott le az egész országot érinto ún. ÜRIK projekt keretében. Ennek

részeként a paksi alállomáson az egy új számítógép alapú mérés és adatgyujto rendszert (SCADA) telepítettek. Az ún. HIR rendszer [160] hálózati kapcsolattal éri el az alállomást közvetlenül muködteto egyedi berendezéseket (RTU), információt továbbít az atomeromu blokki számítógépeinek és más szoftver alapú szabályozók számára, operátori felületet biztosít az üzemeltetoknek és 24 órás üzemben archiválja az alállomás minden digitális és analóg adatát.



29. ábra A paksi alállomás topológiájának G2-beli reprezentációja

A 29. ábra mutatja be az alállomást, amely 5 db 400 kV-os, 15 db 120 kV-os és 1 db transzformátor mezobol áll. Az ábrán a jobb alsó sarokban látható 4 db 400 kV-os mezore érkezik a blokki generátorokból a gépvezetéken keresztül az áram, és a mezok másik oldalán az országos gerinchálózathoz tartozó távvezetékeken (Martonvásár, Sándorfalva stb.) keresztül hagyja el az alállomást. Az ötödik 400 kV-os mezon (alul középen) keresztül jut el az áram a transzformátorokra, amelyek 120 ill. 18 kV-ra transzformálják a feszültséget. Innen kerül a 4 db ún. 120 kV-os transzformátor mezon (az ábrán középen) keresztül a 120 kV-os belso sínekre, ahonnan a kimeno 120 kV-os mezon keresztül a környéki városokba vezeto távvezetékekre (pl. Kalocsa, Dunaújváros stb.). 3 db 120kV-os kimeno vezeték a blokkokhoz vezet vissza, és arra szolgál, hogy az atomeromuvi blokk újraindulásához szükséges villamos energiát tudja szolgáltatni.

Az egyes mezok kapcsolóelemei az alábbiak: (1) a megszakítók, amelyek a nagyfeszültségu áram kapcsolására szolgálnak (az ábrán négyzetek); (2) a vonali szakaszolók, amelyek a feszültség alatti és feszültségmentes részek tartós elválasztását biztosítják (az ábrán körök); és (3) a földelo szakaszolók, amelyek segítségével egy adott mezorész leföldelheto (az ábrán körök + földelési jel). Az ábrán a sötét (a képernyon piros) színu készülékek vannak bekapcsolva, a világosak (a képernyon zöldek) pedig kikapcsolva.

A SCADA rendszer a korábbi megoldással ellentétben megteremtette azt az adatgyujtési és feldolgozási hátteret, amellyel lehetové vált az alállomás tranziens eseményeinek pontos (ms-os) kiértékelése, az egyes veszélyes állapotok idoben történo felismerése, valamint a nagyfeszültségu kapcsoló berendezések élettartamának a számítógépes felügyelete. Ezért döntött úgy az eromu vezetése, hogy a mindenképpen kialakítandó SCADA rendszerhez csatlakozva egy intelligens tanácsadó rendszert [15, 30] is telepíttet az alállomásra az alábbi követelményekkel:

- Az intelligens tanácsadó rendszer a technológus mérnöki tudásnak megfelelo gyors kiértékeléseket biztosítsa a kezeloknek veszélyes üzemállapotban.
- Támogassa a tanácsadó rendszer a karbantartás tervezhetoségét a berendezések pontos muködési történetével, az elhasználódásának a becslésével.
- Biztosítsa védelmi kapcsolások muködése esetében az alállomás gyakorlati muködésének összehasonlítását az elvi tervekkel.
- Támogassa az üzemeltetoket a szokásos kapcsolási szekvenciák megtervezésében.

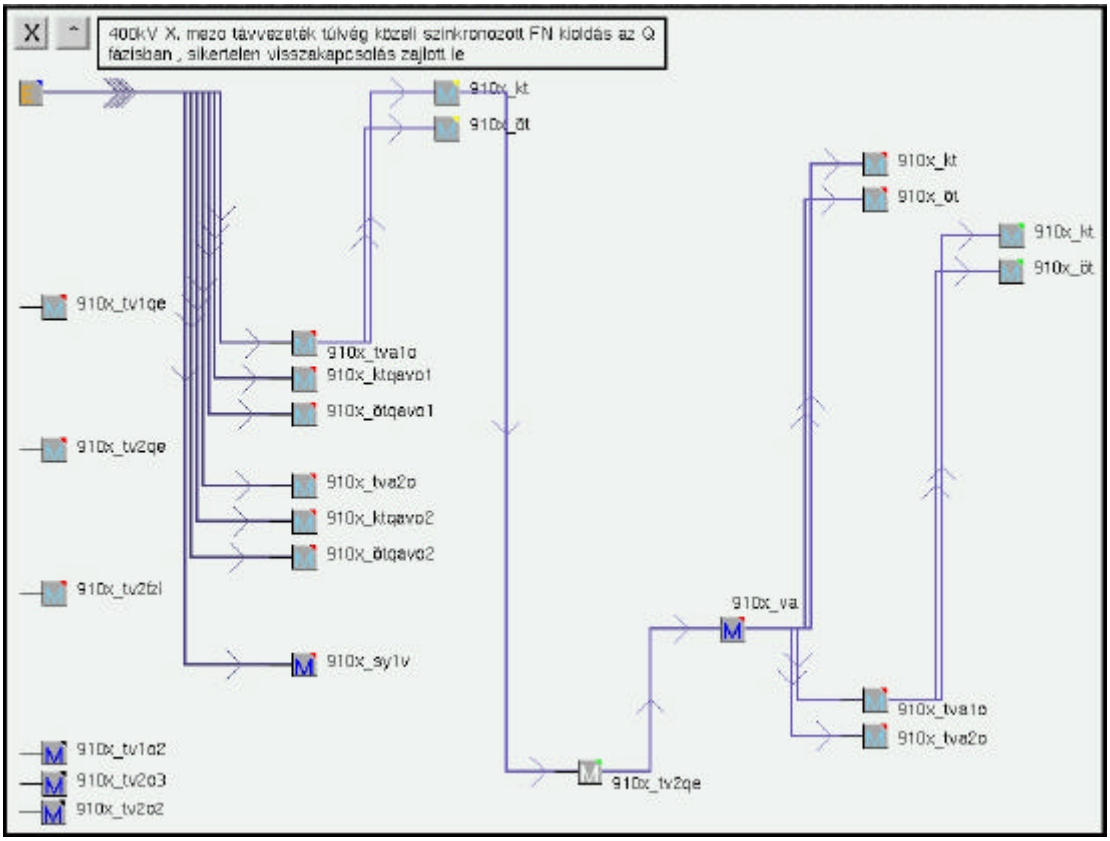
Az intelligens tanácsadó rendszer követelményrendszerének (több, bizonyos értelemben független intelligens funkció) és a muszaki hátterének (adatelérések, hardver-szoftver környezet) az elemzése vezetett arra a felismerésre, hogy a korábban ismertetett KSC koncepció ebben a nem gépész, hanem villamos technológiai problémában is sikerrel alkalmazható. A sokféle funkció közös implementálására kínálkozott jó eszköznek a Gensym G2 nevu intelligens szoftvere [65].

A KSC-ben a jellegükben nagymértékben különbözo tanácsadási funkciókat egységes elvek alapján lehetett kezelni, nyitottan arra, hogy adott esetben egyiket-másikat új algoritmusra áttérve ki lehessen cserélni. A specifikációban megadott intelligens modulok elemzése után egyes modulok összevonásra kerültek, másokról kiderült, hogy az adott specifikáció mellett nem igényelnek MI támogatást, míg olyan önálló funkciójú modulok is felbukkantak, amelyeket a másik modul következtetéseihez kellett kifejleszteni, noha eredetileg a specifikációban nem szerepeltek. A HIR-G2 rendszerben végül az alábbi modulok lettek definiálva:

- A védelmi kiértékelés [31] begyujti egy-vagy több készülék kapcsolásakor fellépo jeleket, kiszuri az üzemi kapcsolásokat (átadva adatait a készülékdiagnosztikának), majd az esetleg fennmaradó jeleket mintaillesztéssel összehasonlítja a paksi technológus mérnökök által definiált védelmi mintatár adataival. A mintatárban a jelekhez prioritások, opcionálisan két jelhez logikai idokapcsolatok tartoznak, valamint mintát kizáró jeleket is lehet definiálni. A mintaillesztés algoritmusa "jutalmazza" vagy "bünteti" a pontosan illeszkedo, a felesleges vagy a meglevo, de az idointervallumból kieso jeleket. A legjobban illeszkedo mintát adja át a kezelonek. Ez az információ lényegesen több annál, amit a SCADA rendszer biztosít, hiszen ott annyi információt lát a kezelo, hogy egy vagy több védelmi jel fellépett, de nincs információ a jelek közötti összefüggésekrol.

  A technológus által definiált védelmi minták egy megfelelo módon specifikált Excel fájlban állnak a HIR-G2 rendelkezésére. Innen a tanácsadó rendszer automatikusan beolvassa, majd minden egyes mintából egy-egy gráfot generál. Egy minta reprezentálhat egyszerre több védelmi eseményt is, mert több mezore (X) és fázisra (Q) vonatkozhat (ún. metaminta). A 30. ábra bal felso sarkában

látható kis kocka - ahonnan sok nyíl indul ki - jelképezi a felismerendo eseményt. A nyilakkal összekapcsolt jelek között áll fenn idobeli kapcsolat. Az egyéb esetekben csak a jel megléte vagy hiánya számít.



30. ábra 400 kV X mezo távvezeték túlvég közeli szinkronozott FN kioldás a Q fázisban, sikertelen visszakapcsolás zajlott le

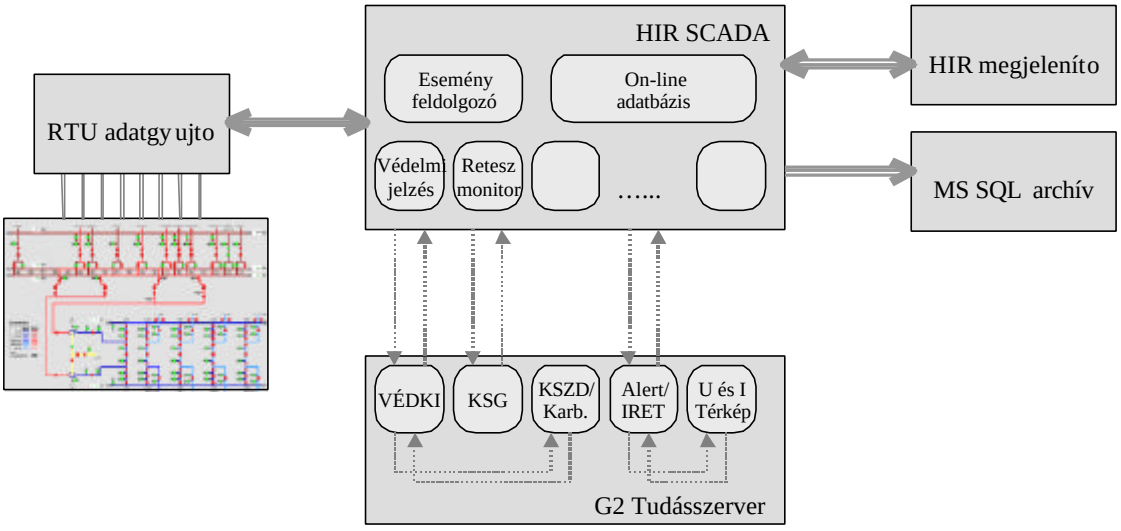
- A kapcsolási sorrend generálás [32] segítségével a kezelo egy elore definiált ún. távmondat gyujteménybol kiválasztott kapcsolási feladathoz tartozó kapcsolási listát kap kézhez. A rendszer az alállomás pillanatnyi állapotából kiindulva kiszámítja a választott távmondatnak megfelelo kapcsolási végállapotot, majd legenerálja a kapcsolások sorrendjét. Az algoritmus minden elemi kapcsolás elott megállapítja, hogy az alállomáson éppen melyik készülék kapcsolható azok közül, amelyeket kapcsolni kell még a végállapot eléréséhez. Ezt oly módon teszi, hogy a kapcsolók fiktív állásjelzéseihez tartozó reteszegyenleteket számolja ki. Majd ezek közül egy szabályhalmaz következteti ki, hogy melyik az adott lépésben a "leginkább" kapcsolandó. Ez a felépítés teszi lehetové, hogy az elektrotechnikai szabályok mellett (pl. áram csak megszakítóval kapcsolható) a helyi üzemvitel egyedi szabályait is figyelembe vegye a rendszer. A távmondat jellegének megfeleloen (ki vagy bekapcsolás; "világosban" vagy "sötétben" áttérés) kell az ún. "off" vagy "on" típusú szabályhalmazt elohívni.

- A készülékdiagnosztika és a karbantartás ütemezés támogatás két olyan funkció, amely egyetlen modulban lett implementálva. A SCADA rendszertol kapott jelek (a készülék állásjelzés változása, kapcsolási parancs) alapján kiszámolja a kapcsolási idejét. A kapcsolás körülményei (üzemi, védelmi) alapján becsüli a

kapcsolás során fellépett terhelést. A kapcsolási ido és a terhelés statisztikai feldolgozása - amely jelenleg nem tartalmaz MI megoldást - támogatja a technológust a karbantartás alkalmával, hogy mely készülékeket kell mindenképpen karbantartani, esetleg kicserélni.

- Az alert állapot felismerés [32] és intelligens retesz felügyelet olyan veszélyes alállomási topológiákat, helyzeteket detektálnak, amelyeket a kezelo nem feltétlenül vagy nem idoben vesz észre. Az alert adott alállomási topológiák és szekunder jelek (pl. nyomásérték csökkenés) együttes meglétekor jelez, ha ezek közös hatása veszélyt rejt magában. (Pl. ha csak egyetlen áramút van bekapcsolva és azon a szakaszon valamely megszakító gáznyomása nem kielégíto.). Ugyanez a funkció egy transzformátor melegedési modell alapján felügyeli a két transzformátorban az olaj homérsékletét, hogy kello idoben (fél órával korábban) lehessen figyelmeztetni a kezelot, ha az 80 fok fölé emelkedne. Az intelligens retesz az egyes ágakban folyó áramok és fellépo feszültségek alapján vizsgálja az egyes készülékek kapcsolhatóságát. Figyelmezteti a kezelot, ha valamely készülék nem kapcsolható, bár engedné a SCADA rendszerbe beépített logikai egyenleteken alapuló ún. hardver retesz, amelynek eredményét a kezelo közvetlenül látja, ha kapcsolni akar egy készüléket.

- A feszültség- és áramtérkép kiszámítása úgy történik, hogy az alállomáson mért U és elojeles I értékeket "elterjeszti" - az áram esetében a Kirchoff törvényeket használva - az egész topológián. Így ez a funkció fogja az ún. sánta üzemet jelezni a kezelonek, amikor a három fázis között jelentos áram vagy feszültség különbség van. A két térképet a kezelo megjelenítojére is elküldi a rendszer, de a térképeket elsosorban az alert és intelligens retesz funkció használja.



31. ábra KSC a Paksi Atomeromu tanácsadó rendszerében

A 31. ábra a megvalósított HIR-G2 tanácsadó logikai kapcsolódását mutatja a HIR rendszerhez. Néhány funkciót a SCADA rendszer indít, és vannak olyanok is, amelyek pedig egy másik funkció kiszolgálására kezdenek el futni.

Az adatkapcsolat szintjén egy három részbol álló közös adatinterfész köti össze a SCADA rendszert a KSC-vel. Egy bemeneti FIFO csatornán (1) keresztül kapja a HIR-

G2 rendszer a fontos digitális jelek változását. Ezek a jelváltozások indítják a tudásszerver muködését. Majd a HIR-G2 adattömbökben (2) kérdezi le közvetlenül az analóg jeleket, amelyek szükségesek a következtetések végrehajtásához. A döntéseit, figyelmeztetéseit egy másik, kimeneti FIFO-n (3) keresztül üzeni meg a SCADA rendszert használó kezelonek.

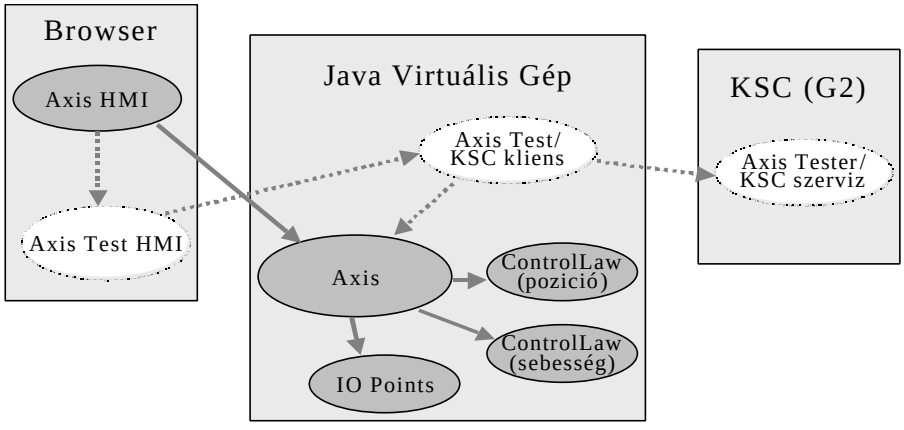
A KSC adott ciklus idovel olvassa a SCADA rendszer off-line adatbázisát, ahol a készülékek alapadatai (pl. logikai reteszegyenletek, kapcsolási idok referencia értékei) találhatók. Ha ez az adatbázis módosul - azaz változik a verziószáma - a HIR-G2 automatikusan áttölti az összes referencia adatát. Eközben futási idoben újragenerálja azokat a program részeit, amelyek az egyes kapcsolókészülékekhez tartozó logikai reteszegyenleteket képesek kiszámolni, amiket a rendszer a kapcsolási sorrend generálása során használ.

A HIR SCADA rendszer 1999, a HIR-G2 rendszer 2000 eleje óta fut a Paksi Atomeromu 400/120-as villamos alállomásán, 2001 elején indult el az a Terminál Server program, amely az egész eromuvi intranet-en láthatóvá tette a tanácsadó rendszer muködését.

8.4 INTELLIGENS CNC PROTOTÍPUSA KSC SEGÍTSÉGÉVEL

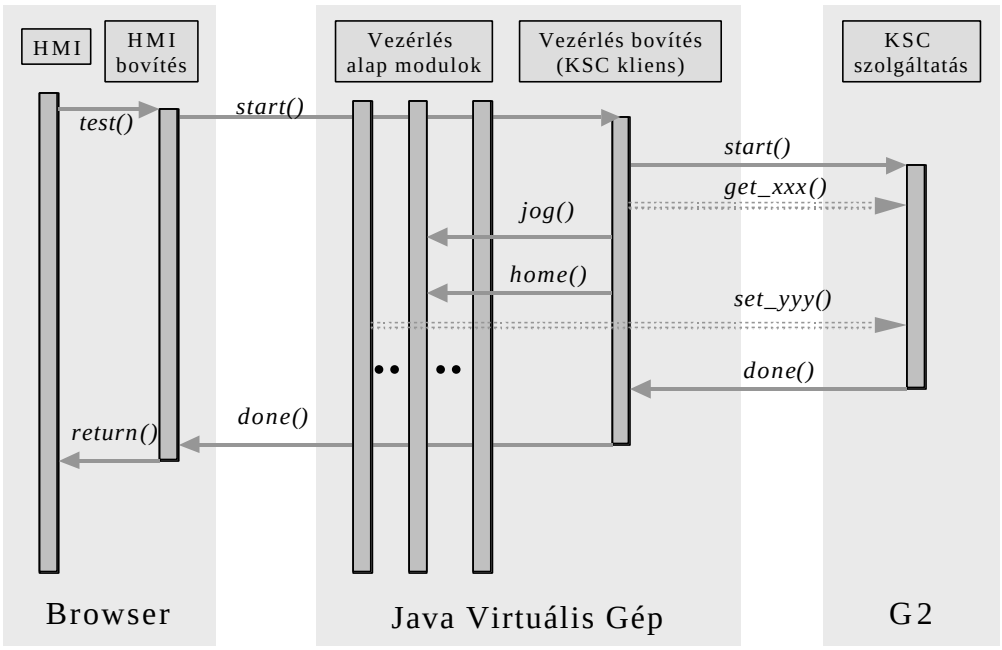
A KSC-n alapuló intelligens CNC prototípusához az elozményeket - elsosorban tanulságaival - a SzTAKI-ban Drozdik Szilveszter által kifejlesztett OSACA alapú DAC rendszer jelentette [52]. Itt a Corba környezetben sikeresen lehetett különbözo szoftverelemekbol (C, Java) álló komponenseket összeépíteni. Ugyanakkor a 4.2.4 alfejezetben bemutatott analízis miatt az intelligens CNC prototípusánál úgy döntöttem, hogy az [19, 33] nem OSACA, hanem OMAC referencia architektúrán alapuljon.

Az OMAC API specifikációban [130], annak a 2.3.2 alfejezetében szerepel példaként egy kézi vezérléssel muködo egytengelyu berendezés, amely alkalmas alapállásba küldésre (*homing*) és finommozgatásra (*jogging*). Az architektúra pozíció és sebesség szabályozást támogat. Az alacsonyszintu IO kezelését (tachométer, home és végállás kapcsolók) úgy oldottam meg, hogy a szimulált futás után az Advantech PCL832-es Motion Control kártyák segítségével a labor marógépéhez lehessen illeszteni a vezérlést.



32. ábra Egyetlen tengelyt tesztelo intelligens CNC prototípusának architektúrája

A 32. ábra sötét ellipszisei mutatják a kiindulást jelento egytengelyes vezérlés moduljait. A tudásszerver ebben a prototípusban a tengely intelligens tesztelését, beállítását támogatja. A kezelo megadhatja (*Axis Test HMI*), hogy alapállásba küldést, végállást, akar-e kalibrálni, beszabályozni (a lehetséges tesztek típusa természetesen bovítheto). Ez indítja a vezérlésben futó tesztelo modult (*Axis Test*) és a tesztelés folyamán ez a modul irányítja tengelyt. Azt azonban, hogy milyen pozícióba, milyen sebességgel küldje a tengelyt a KSC-n futó modul (*Axis Tester*) segítségével számolja ki, vagyis a két modul együttesen biztosítja az intelligens tesztelést. A KSC-ben futó *Axis Tester* szabályai függetlenek konkrét tengely adataitól, szervizként szolgáltat tesztelési javaslatokat és értékeli az eredményt. A példa meglehetosen egyszeru, célja ugyanakkor a KSC koncepció demonstrálása CNC vezérlo környezetben. A 33. ábra a tengely tesztelo szoftver muködésének idobeli lefutását mutatja be.



33. ábra Egy KSC szerviz idobeli lefolyása

A 33. ábra egyben a prototípus szoftver környezetét mutatja, a HMI és a vezérlo programjai Java 2-ben készültek (Sun jdk 1.3 fejlesztoi eszközzel), a HMI részek önálló java programként ill. megfelelo böngészoben vagy a *jdk appletviewer*-rel appletként is futtathatóak. A KSC G2-ben lett implementálva [17]. A modulok közötti kommunikáció a DAC-hoz hasonlóan Corbában lett megvalósítva. Intelligens gyártásban mások is alkalmazták már a CORBA környezetet ágens jellegu feladatokban (pl. [114]). A tengely (Axis) modul a 11.3 függelékben bemutatott egyszerusített OMAC modellt követi. A tesztelo két modulja számára két egyszeru interfészt kellett definiálni, amelyek a 11.6 függelékben láthatóak. Az egyik (*AiAxisTest.idl*) specifikálja a tesztelohöz kapcsolódó felhasználói felület interfészét, a másik (*KscTester.idl*) pedig a KSC felületét, amelyen keresztül a vezérloben futó tesztelo program éri el a szabályalapú tesztelo szervizt. Mindkét modul a Corbán keresztül el tudja érni a tengely modul különbözo interfészeit is.

9 ÖSSZEFOGLALÁS

Értekezésemben bemutattam kutatási munkámat, amelyben a gépvezérlések területén meglévo egyes problémákra kerestem válaszokat a nyílt rendszerek és a mesterséges intelligencia módszereinek a felhasználásával. Nem gondolom, hogy eddigi munkám eredményeképpen ezen a területen véglegesen kielégíto megoldásokat tudtam volna adni, bár az állandóan megújuló lehetoségek és problémafelvetések miatt erre elvileg sem látok esélyt. Ugyanakkor azt hiszem, hogy a tézisekben megfogalmazott állításaim segíthetik, hogy jobban használható, integrálható és "intelligensebb" berendezések jelenhessenek meg a gyártásban.

Az értekezés végén összefoglalom az eredményeimet és kimondom tézisek formájában, majd röviden kifejtem a munka folytatására vonatkozó elképzeléseimet.

9.1 TÉZISEK

1. Tézis - Intelligens kommunikáció logikai szintjei

A gyártórendszerek vezérlésének fejlesztése munkahelyem, a SzTAKI CIM Kutatólaboratóriumának egyik központi feladata volt. Az autonóm komponensek kommunikációjának kutatása során jelentos tapasztalatot gyujtöttünk össze.

> *Bevezettem gyártórendszerekben a logikai kommunikáció három szintjét, amelyekrol megmutattam, hogy megfelelnek a hagyományos tudás-feldolgozási kategóriáknak. Tudásbázisú szimulációval illusztráltam, hogy az egyes kommunikációs üzenetek alapján hogyan elemezheto egy gyártórendszer elosztott intelligens muködése.*

Sem az olyan gyártórendszerekben, ahol több intelligens berendezés található és ezek egymással is kommunikálnak, sem az utóbbi években elotérbe került intelligens ágens alapú rendszerekben nem megoldott a különféle üzeneteknek a futás közbeni elemzési lehetosége. A három-szintu osztályozás (vezérlési adatok, tudásgyujtés és tudás megosztás) segít a kooperáció szintjének feltérképezésében. Ezek a kategóriák lényegében egy az egy-es megfeleltetésben vannak a lehetséges fogalomalkotási típusokkal, és lazább kapcsolatban a különbözo szintu protokollokkal.

A különféle üzenetek - éppen a kategóriák egyszerusége miatt - tartalmukban jól elkülöníthetoek. Egy tudásbázisú környezetben felépített gyártórendszer szimuláció, amely futása közben generálja az egyes készülékek egymás közötti üzeneteit, alkalmas keretet biztosít az osztályozás elvégzésére és a különféle üzenetek között meglévo pillanatnyi arányok, kapcsolatok vizsgálatára.

2. Tézis - Nyílt gépvezérlések

Az elmúlt években bekapcsolódtam jelentos világméretu nyílt gépvezérlés fejlesztési munkákba. Így társszerzoként részt vettem az európai OSACA nyílt vezérlés referencia kézikönyvének kidolgozásában és az amerikai OMAC nyílt vezérlés egyik konkrét megvalósításának kritikai elemzésében.

Metodikát dolgoztam ki önálló nyílt gépvezérlések értékelésére programozói interfészeik (API), referencia architektúráik és infrastruktúráik (hardver-szoftver környezet) elemzésével. Ezt alkalmazva összehasonlítottam a három legfontosabb nyílt gépvezérlési kezdeményezést (OSACA, OMAC, OSEC). Bebizonyítottam, hogy ezek egymással nem kompatibilisek, hogy jelen formájukban ezek alapján nem lehet egy közös általános nyílt gépvezérlést kialakítani. Javaslatot tettem az OMAC alapján való továbblépésre, eközben megoldást adtam az OMAC legnagyobb hátrányának a kezelésére - a bonyolult programozói felületének az egyszerusítésére -, hogy a gyakorlatban használható megoldást kapjunk.

A különféle nyílt gépvezérlési iniciatívák gyakorlatilag ugyanazon felismerések alapján születtek meg és céljaikban is lényegében megegyeznek. Ennek ellenére részletes kiértékelésem bebizonyította és példákkal igazolta, hogy milyen sokféle és szerteágazó inkompatibilitás áll fenn a kidolgozott megoldások között.

Az értékelésben egyértelmuen a legmegalapozottabbnak az OMAC iniciatíva bizonyult, így ennek mentén javaslom a további fejlesztéseket. Felismertem, hogy a kevés OMAC implementáció legfobb oka programozói felületének jelenlegi túlzott bonyolultsága. Erre egy olyan megoldást adtam, amely az egyes OMAC modulok interfészeinek csak egyes elemeit tartja meg kötelezonek, a többit opcionálisnak kezeli. Ezt részletesen kidolgoztam a tengelyvezérlo modul esetében. Az egyszerusítés alapelveként meghatároztam, hogy a minimum készlettel a modulok alapfunkciói megvalósíthatóak maradjanak és a modulok továbbra is kompatibilisek legyenek belso muködésükre specifikált véges állapot gép (FSM - final state machine) modellekkel.

3. Tézis - Nyílt ívhegeszto robot

Ívhegeszto robotok gyakorlati használatát nagymértékben megkönnyítik a technológiatervezés és végrehajtás feladatát együttesen (integráltan) kezelo rendszerek. Ilyen off-line programozói rendszert fejlesztettünk az EU Copernicus program PROARC (CAD-Based Programming System for Arc Welding Robots) projektjében alacsony költségu PC-s környezetben. A szélesebb köru felhasználhatóság miatt törekedtünk a rendszer nyílt kialakítására.

3.1 Definiáltam és megvalósítottam egy STEP alapú, bovítheto ívhegeszto robot nyelvet, amely alkalmas kissorozatú, egyszeru mozgásokból álló hegesztések leírására és integrálva tartalmazza mind a varratonkénti, mind a varratcsoportonkénti hegesztési utasításokat.

A 90-es években a STEP-beli leírás a gyártás különbözo feladataiban elfogadott és szabványos eszközzé vált. Ezt felismerve az egyszeru (egyenes, kör és ptp) mozgásokból álló ívhegesztésekre egy olyan objektum orientált programleírást definiáltam, amely rugalmasan bovítheto és forrásában is olvasható. Négy alapmuveletbol - pozicionálás, hegesztés, off-line keresés és különleges muveletek (pisztoly tisztítás, pisztoly csere) - állítható össze a robotprogram. A robotprogramot úgy alakítottam ki, hogy az a hegesztési paramétereket is tartalmazza (feszültség, áram, huzalsebesség), támogassa az ívhegesztési technológia szokványos muveleteit (pl. rezegtetés, a varrat végén keletkezo kráter feltöltése rövid visszamozgással) és a hegesztés során K és V varratok esetében használt legtipikusabb keresési muveleteket.

3.2 Kidolgoztam és sikeresen implementáltam az MMS robotszabványon alapuló ívhegeszto robot virtuális gyártóberendezést, az ún. AWR-VMD-t (Arc Welding Robot - Virtual Manufacturing Device).

Az MMS protokoll lehetové teszi a virtuális gyártóberendezésekben különféle objektumok egyedi specifikációját. Az ún. RobotVMD modellt fejlesztettem tovább azzal, hogy az ívhegeszto berendezés jellemzoit egyetlen tartományba (domain-be) foglaltam össze. Ezt a tartományt a robot virtuális gép részeként definiáltam, de a modell alkalmas arra is, hogy akár egy önálló virtuális gépként szerepeljen.

A STEP alapú programnyelvet és az AWR-VMD-t sikeresen alkalmaztam egy off-line robotprogramozói munkaállomás fejlesztésében, az eredmények nyomán egy KUKA ívhegeszto robotot lehetett sikeresen nyílt hálózatban használni.

4. Tézis - Intelligens megoldások nyílt vezérlésekben

A 3. tézishez kötheto ívhegeszto robot programozói rendszer eredményeinek MI eszközökkel történo továbbfejlesztése olyan intelligens funkciók specifikálását tette lehetové, amely a korábbi projektben feltárt korlátokat jelentosen meghaladta. A VMD modell létrehozása irányította figyelmemet az MMS objektum orientált megközelítésében rejlo további lehetoségekre.

4.1 Felismertem és igazoltam, hogy az MMS protokoll alkalmas eszköz tudásbázisú cellavezérloben a vezérlendo berendezések belso modellezésére is. Elsoként fejlesztettem ki MMS hálózatban muködo intelligens cellavezérlo prototípusát robotcella irányítására. Erre a környezetre alapozva definiáltam ívhegeszto robotok off-line programozói munkahelyének olyan tudásbázisú bovítését, amely a hegesztés tervezéstol a gyártásig egy rendszerben támogatja a felhasználót.

Az MMS protokollon alapuló virtuális gyártóeszköz modell egy adott gyártóberendezést reprezentál, amely elegendo egy tudásbázisú cellavezérlo belso adathalmazában a berendezés definiálására. Egy belso, az intelligens keretrendszerben implementált MMS könyvtár szolgáltatásait használva a programok letöltése, a távoli program vezérlése vagy a legfontosabb adatok lekérdezése automatikusan végrehajtódnak, amikor a cellavezérlo belso következtetései során vagy a kezelo kezdeményezésére erre szükség van.

A technológiai adatok adatbázisban tárolt default értékei helyett szabályokban és frame-kben lehet leírni a szakértoktol összegyujtött hegesztési ismereteket. Az így megtervezésre került rendszer komplex tudásbázisa, amely többféle tudásmodulból áll, a CAD rajztól kiindulva a gyártásig minden fázisban támogatja a hegesztési feladat végrehajtását.

A cellavezérlési feladatok után felismertem, hogy a nyílt gépvezérlések kifejezetten alkalmas alapot jelentenek MI eszközök vezérlésekbe integrálására.

4.2 Elsoként alkalmaztam OSACA környezetben MI módszereket és kifejlesztettem egy G2 alapú, ember-gép kapcsolattal rendelkezo CNC prototípusát OSACA platformon, amellyel a kezelo beállításait lehet támogatni vagy korrigálni egy háttérben futó szabályhalmaz segítségével.

Az MI eszközök vezérlésekbe történo beillesztésének komoly korlátját jelenti a vezérlések zártsága, azaz, hogy tipikusan csak a felhasználói felület nyílt. A nyílt gépvezérlések pontosan specifikált modulhierarchiája segítségével azonban egy olyan általános környezetet tudtam egyszeruen kialakítani, ahol egyes modulok MI alapúak.

Igazolásul az OSACA demonstrációs szoftver egy modulját, a HMI-t (Human Machine Interface) emeltem át MI alapú G2-es környezetbe. Ehhez egy speciális gateway szoftvert is ki kellett fejlesztenem az OSACA platform és a G2 között. Az adott prototípus korlátai között a G2 tudásbázisa a kezelo beállításait módosította (pl. adott pozíciókban a tengelysebesség limitálása).

5. Tézis - Vezérlések tudásszervere (KSC)

Noha a CNC-k az elmúlt 10 évben jelentosen fejlodtek, az intelligens CNC-k megjelenése továbbra is várat magára. Több szempontból elemeztem az intelligens CNC-k helyzetét, vizsgálataimban az elvárások és a kutatási eredmények áttekintésével eljutottam a jelenlegi korlátok és lehetoségek összegyujtéséhez Eközben a gépvezérlések belso, strukturális adaptivitását három szinttel (Nincs, Korlátozott, Teljes) jellemeztem.

> *Intelligens CNC-k létrehozása céljából, de általánosabb felhasználási lehetoséggel bevezettem és megvalósítottam a vezérlések tudásszervere (KSC) koncepciót, amely alkalmas nyílt gépvezérlések MI eszközökkel való kiegészítésére. A KSC koncepciót sikeresen igazoltam egy más jellegu - villamos energia ipari - alkalmazásban.*

Meghatározásom szerint a vezérlések tudásszervere (KSC) olyan hálózati eroforrás, ami intelligens algoritmusok hatékony futtatását végzi és így egyéb rendszerek számára az intelligens muködés lehetoségét szolgáltatja. Ma már egyre több MI környezetnek van komponens alapú interfésze is (pl. Corba), valamint hasonló tulajdonságokkal rendelkezo nyílt vezérlési architektúrák is ismertek. Azt állítom, hogy lehetoség van a KSC kialakítására CNC környezetben és így olyan infrastruktúra építheto, amely kiválóan alkalmas intelligens CNC-k létrehozására. Továbbá a KSC lehetové teszi, hogy egy nagyobb környezetben (pl. muhely) több egymástól függetlenül futó berendezés is ugyanazt a KSC-t használja önálló, vagy esetleg részben összefüggo MI alapú feladatok végrehajtására, ezzel költségtakarékos megoldást biztosítva. Elemeztem a KSC és az intelligens ágens struktúrák közötti hasonlóságokat és különbségeket.

A KSC koncepcióm igazolásául egy egyszerusített OMAC alapú (2. tézis) egytengelyes vezérlést kapcsoltam G2 környezethez Corba felületeken keresztül. Az intelligens modul a tengely teszteléséhez használja az MI eroforrásokat.

A KSC-t ipari alkalmazásban is sikeresen alkalmaztam és így a koncepció muködoképességét bebizonyítottam a Paksi Atomeromu 400/120 kV-os villamos alállomásán kialakított tanácsadó rendszerben, amely többféle intelligens funkciójával támogatja az ügyeletes kezeloket és a technológus mérnököket.

9.2 A TOVÁBBLÉPÉS LEHETOSÉGEI

Szeretnék további elemzéseket és szimulációs vizsgálatokat végezni az intelligens kommunikáció logikai szintjei témakörében, esetleg valódi rendszerek adatait elemezni, másrészt azt megvizsgálni, hogy az üzenetekbol kiindulva pontosan milyen (statisztikai) jellemzok segítségével lehet hatékonyan vizsgálni az elosztott intelligencia hatását a gyártórendszerben.

A közeljövoben elottem álló izgalmas kihívásnak látom egy valódi szerszámgép teljes gépvezérlésének a megalkotását az ún. egyszerusített OMAC specifikáció szerint, ill. ezután egy ilyen berendezés kiegészítését néhány intelligens modullal, amelyek KSC-hez kapcsolódnak. Ide tartozik, hogy foglalkoztat egy STEP-NC nyelvet érto és ez alapján muködo CNC prototípusának a megalkotása ebben a környezetben. A KSC és a STEP-NC együttesében nagyon komoly távlati lehetoségeket látok.

Ezután szeretném egy KSC-n alapuló intelligens CNC prototípusát elkészíteni ezekbol a modulokból és egy valódi szerszámgéphez kapcsolni. A prototípus nyomán lehetne kidolgozni azt az általános metodikát, ill. referencia architektúrát, hogy egy intelligens CNC-ben milyen modulok legyenek, ezek közül melyek opcionálisak, meg lehet-e adni ezeknek valamiféle általános modul interfészét. Később ezt a metodikát lehetne kiterjeszteni robot és cellavezérlokre is.

Komoly kutatási és alkalmazási lehetoségét látom a KSC-nek a SzTAKI, GE Tungsram, BME és ME által a Széchenyi pályázaton frissen elnyert "Digitális gyár" munkában, ahol különféle tudásmodulokat kell a lámpagyár megoldatlan problémáira felépíteni. Úgy látom, hogy ez a feladat a cella szintu feladatokat támogató KSC felé jelent majd továbblépést.

1-2 éves perspektívában a konkrét ipari alkalmazás területén remélem, hogy további tanácsadó rendszereket - amelyek KSC elven muködnek - tudunk majd a Paksi Atomeromuben üzembe helyezni. Ezt segítheti, hogy az alállomáson használt tudásszerver környezet (MS Windows NT - operációs rendszer, Intellution FIX - SCADA rendszer) megegyezik az atomeromu blokki számítógépein lévovel.

Az alállomási tanácsadó rendszer esetében pedig egy beadott ESPRIT pályázat (ISISS) támogatásával kevésbé "Paks specifikus" megoldások kidolgozása történne meg. Ez magában foglalná a legizgalmasabb "védelmi kiértékelés" funkció esetében egy modell alapú megoldás kifejlesztését, ahol a G2 modellezési képességei lennének fokozottan kihasználva. Érdeklodést tapasztaltunk a bolgár villamosmuvek részérol is.

Tennivaló és nyitott kérdés tehát boven akad, amelyek további izgalmas kihívást jelentenek számomra.

10 IRODALOMJEGYZÉK

10.1 A SZERZONEK AZ ÉRTEKEZÉSBEN HIVATKOZOTT PUBLIKÁCIÓI

1. Amezaga J., Barg J., Brühl J., Lutz P., Nacsa J., Pohl M., Sozzi M., Wälde K., Zulauf P.: OSACA Handbook, OSACA Association, Stuttgart, Germany, 1997
2. Haidegger G., Nacsa J.: Shop-Floor Communication With OSACA-Compliant Controllers, In: Proc. of the IEEE Workshop on Factory Communication Systems, Barcelona, Spain, 1997 Oct. 1-3, pp. 355-362
3. Haidegger G., Nacsa J., Bausz I.: Applying SERCOS Industrial Control Network for OSACA Based CNC-s, Proc. of Symposium on Fieldbus Systems and Appilication Technics, Budapest, Hungary 1997 Febr. 17-19, pp. 85-92
4. Kovács, G. L., Haidegger G., Nacsa J.: Some Problems of Intelligent Control of Open CNCs and Manufacturing Cells, plenary paper, Preprints of the IFAC Symposium on Artificial Intelligence in Real-Time Control (AIRTC), Budapest, 2000 October 2-4, pp. 21-28.
5. Kovács G., I. Mezgár, S. Kopácsi, J. Nacsa, P. Groumpos: A hybrid simulation-scheduler-quality control system for FMS. In: Mechatronics. The basis for new industrial development. Proceedings of the joint Hungarian-British international mechatronics conference. Budapest, 1993. Southampton, Computational Mechanics Publ., 1994. pp. 655-662.
6. Kovács G.L., Mezgár I., Kopácsi S., Gavalcová D., Nacsa J: Application of artificial intelligence to problems in advanced manufacturing systems, Computer Integrated Manufacturing Systems, Vol. 7 (1994), No.3
7. Kovács G.L., Nacsa J.: Towards intelligent and open control of manufacturing systems. In: 3rd IFAC/IFIP/IFORS workshop on intelligent manufacturing systems. IMS'95. Preprints. Vol. 1. Bucharest, IFAC, 1995. pp. 165-169.
8. Kovács, G.L., Nacsa, J.: Application of Knowledge Based Control for an Integrated Robotized CAD/CAM Arc Welding System, Proc. of the 12th Int. Conf. on CAD/CAM, Robotics and Factories of the Future, London, UK, 14-16. August, 1996, pp. 381-386.
9. Kovács, G.L., Nacsa, J.: Some Communication Problems of KB Controlled Manufacturing Systems, Proceedings of the 27th International Symposium on Industrial Robots, 6-8 Oct. 1996, Milan, Italy pp. 829-834.
10. Kovács G.L., Nacsa J.: Some Communication Problems of KB-controlled Manufacturing Systems. Engineering Application of Artificial Intelligence (Elsevier Science Ltd.), Vol. 10, No. 2, pp. 225-230, 1997.
11. Kovács G.L., Nagy G., Gavalcová D., Nacsa J.: Interfacing G2 Expert System Shell with MMS Protocol, Gensym User Society Worldwide Meeting, Washington D.C., U.S.A., May 4-6, 1994
12. Kopácsi S., G.L. Kovács, J. Nacsa, G. Haidegger, S. Drozdik et al.: Assessment of OAC/MOS Ver. 1.0 Controller, Report, MTA SzTAKI, 1997
13. Monostori L., Markos S., Krämer S., Nacsa J., Szöllõsi G.: Neural networks for modeling and monitoring of manufacturing processes; milling, 8th International IMEKO Symposium on Technical Diagnostics, Dresden, Germany, 23-25 Sept. 1992

14. Nacsa, J.: MMS Communication in a Robotized Welding Application, 5th Int. Workshop on Robotics in Alpe-Adriaa-Danube Region, RAAD'96, June 10-13, 1996, Budapest, Hungary, Proceedings Ed. I.J.Rudas, pp.235-238
15. Nacsa J.: G2 Based Advisory System at the 400/120 kV Substation of the Paks Nuclear Power Plant, Preprints of the 2nd Mexican-Hungarian Workshop on Factory Automation and Material Sciences, Queretaro, Mexico, 9-10 March 1999, pp. 109-115; also appeared in Gensym Web Site as a Success Story: http://www.gensym.com/expert_operations/PAPERS/Nacsa.html
16. Nacsa, J.: An Open Communication Interface for Arc Welding Robots. In: Proc. of 7th Int. Conf. on Computer Technology in Welding, San Fransisco, USA, 1997 July 8-11, American Welding Society
17. Nacsa J: Intelligent Machine Tools based on the Gensym's FIDA Technology – A Vision, Gensym User Society Annual Meeting, Barcelona, 2000, CD-ROM
18. Nacsa J.: Comparison of three different open architecture controllers, IFAC MIM, Prague, 2-4 Aug. 2001, (accepted)
19. Nacsa J.: Intelligent Open CNC System Based on the Knowledge Server Concept, IFIP Prolamat, Budapest, 7-10 Nov. 2001 (accepted)
20. Nacsa J.: Logical Communication Levels in an Intelligent Flexible Manufacturing System, IFIP Prolamat, Budapest, 7-10 Nov. 2001 (accepted)
21. Nacsa J., Haidegger G.: MAP activities in Hungary - Setting up a "MAP Training Center", Enterprise Network Event (ENE'92) International Conference of SME, Washington D.C., U.S.A., 16-19 March 1992
22. Nacsa J, G. Haidegger: Built-in Intelligent Control Applications of Open CNCs, Proc. of the Second World Congress on Intelligent Manufacturing Processes and Systems, Budapest, Hungary, 1997 June 10-13., Springer, (Ed. L. Monostori), pp. 388-392
23. Nacsa J., Haidegger G., Kovács G. L.: Intelligent Control Applications of Open CNCs Using G2, Proc. of the Mexican-Hungarian Workshop on Factory Automation and Material Sciences, Budapest, 25-29 May, 1998, pp. 74-82.
24. Nacsa J., Kovács G.L.: Communication problems of expert systems in manufacturing environment, Symposium on Artificial Intelligence in Real-Time Control, Valencia, Spain, AIRTC'94, Oct. 3-5, 1994
25. Nacsa J., Kovács G. L.: Some problems of knowledge based control of manufacturing systems using open communication. In: Applications of artificial intelligence in engieneering X. AIENG '95. Tenth international conference. Udine, 1995. (Eds. R. A. Adey, G. Ryevski, C. Tasso.) Southampton, Computational Mechanics Publications, 1995. pp. 349-356.
26. Nacsa J., Kovács G. L.: A system to assist robotized welding for SME's. In: 3rd IFAC/IFIP/IFORS workshop on intelligent manufacturing systems. IMS '95. Preprints. Vol. 2. Bucharest, IFAC, 1995. pp. 381-386.
27. Nacsa J, G. Kovács: A PC-based OSI System to Assist Robotized Welding, Opening Productive Partnerships: Proceedings of the Conference on Integration in Manufacturing, Vienna, 1995, IOS Press
28. Nacsa, J., Kovács G.L.: An Integrated CAD/CAM System for Robotized Arc Welding, Proceedings of the 27th International Symposium on Industrial Robots (ed. Prof. D. Fabrizi), 6-8 Oct. 1996, Milan, Italy pp. 287-292.
29. Nacsa J., Kovács G. L.: An Integrated CAD/CAM System for Robotized Arc Welding, Studies in Informatics and Control, Vol/8 No/1, March 1999, pp.7-18.
30. Nacsa J., Kovács G. L.: Alert State Detection and Switching Order Generation at a 400/120 kV Substation using Knowledge Server, ISAP2001 Conference. (Intelligent Systems Application to Power Systems), Budapest, 18-21 June, 2001

31. Nacsa J., G. L. Kovács, S. Kopácsi: Application at the 400/120 kV Substation of the Paks Nuclear Power Plant, IEEE, Budapest, Hungary, 1999
32. Nacsa J., G. L. Kovács, G. Szederkényi: Intelligent Alert State Detection and Switching Order Generation at the 400/120 kV Substation of the Paks Nuclear Power Plant, SAFEPROCESS, Budapest, Hungary, 2000
33. Nacsa J., Kovács G.L., Haidegger G.: Intelligent, Open Architecture Controller Using Knowledge Server, SPIE's Int. Symp. on Intelligent Systems and Advanced Manufacturing, 28-31 Oct. 2001, Newton, MA, USA (under evaluation)
34. Peper S., A. Szilvásy, J. Nacsa: PROARC final report, 1996, http://ww.sztaki.hu/sztaki/ake/cim/proarc/proarc-final.ps

10.2 FELHASZNÁLT PUBLIKÁCIÓK

35. Altinas Y, N.A. Erol: Open Architecture Modular Tool Kit for Motion and Machining Process Control, Annals of the CIRP, Vol 47/1, 1998, pp. 295-300
36. Barschdorff D., L. Monostori, Wöstenkühler G.W, Egresits Cs, Kádár B: Approaches to coupling connectionist and expert systems in intelligent manufacturing, Computers in Industry, Vol. 33, 1997, pp. 5-15
37. Basden A: The Istar Knowledge Server, 2000 http://www.basden.u-net.com/pgm/Istar/index.html
38. Bjarnason E., G. Hedin, K. Nilsson: Interactive Language Development for Embedded Systems, Nordic Journal of Computing, 6(1):36, Spring 1999.
39. Bongaerts L., Integration of Scheduling and Control in Holonic Manufacturing Systems, Ph.D. Thesis, K.U.Leuven, PMA 98D11, 1998. http://adhemar.mech.kuleuven.ac.be/~lbongaer/doc/order.html
40. Bongaerts L., H. Van Brussel, P. Valckenaers: Generic Concepts for Holonic Manufacturing, Control,
41. Brill M., Gramm U.: MMS: MAP application services for the manufacturing industry, Computer Networks and ISDN Systems, 1991, 21, pp. 357-380
42. Chao K.M., P. Smith, W. Hills, B. Florida-James, P. Norman: Knowledge Sharing and reuse for engineering design integration, Expert Systems With Applications 14 (1988) 399-408
43. Chaudhri F. V., R. Fikes, P. Karp, J. Rice. OKBC: A Programmatic Foundation for Knowledge Base Interoperability. Proceedings of AAAI-98, Madison, Wisconsin, February, 1998.
44. Cheng, T., J. Zhang, C. Hu, B. Wu, S. Yang: Intelligent Machine Tools in a Distributed Network Manufacturing Mode Environment, Int. J. Adv. Manuf. Technol. (2001) 17:221-232
45. Chrysler, Ford Motor and General Motors. Requirements of Open, Modular Architecture Controllers for Applications in the Automotive Industry, 1994, White Paper - Ver. 1.1. http://www.arcweb.com/omac
46. CommonKADS, http://swi.psy.uva.nj/projects/CommonKADS
47. Cselényi J., T. Tóth: Some question of logistics in the case of holonic production systems, J. of Intelligent Manufacturing (1998) 9, pp. 113-118
48. Daume S., D. Robertson: An architecture for the deployment of mobile decision support systems, Expert Systems with Applications 19 (2000) 305-318
49. Delmia Inc: Robotics Solutions, IGRIP, 2001 http://www.delmia.com/solutions/html/robotics.htm

50. Deng P.S., E.G. Tsacle: Coupling genetic algoritms and rule-based systems for complex decisions, Expert Systems with Applications 19 (2000) 209-218
51. Drozdik Sz.: Presentation of the OSACA demonstration software, Ver. 2.0, OSACA Open Day, Stuttgart, 1998
52. Drozdik Sz: DAC Rendszerterv + DAC Virtuális Gép Rendszerterv + DAC NC kernel rendszerterv, Report, Atysoft - MTA SzTAKI, 1999
53. Dumur D., P. Boucher, J. Röder: Advantages of an Open Architecture Structure for the Design of Predictive Controllers for Motor Drives, Annals of the CIRP, Vol 47/1, 1998, pp. 291-294
54. Egresits, Cs.; Monostori, L.; Hornyák, J.: Multistrategy learning approaches to generate and tune fuzzy control structures and their applications in manufacturing, Journal of Intelligent Manufacturing, Vol. 9, No. 4, August 1998, Special Issue on Soft Computing Approaches to Manufacturing, Chapman & Hall, pp. 323-329.
55. EPRI (Electric Power Research Institute): Introduction to the UCA (Utility Communications Architecture) Version 2.0, Prepared under the Auspices of the Profile Working Group of the MMS Forum, 1997
56. ESS Gmbh, http://www.ess-gmbh.com/index1.htm
57. Erdélyi F.: Gyártórendszerek irányításának hierarchiája, Automatizásás'95, pp. 383-391
58. Eriksson H: Expert Systems as Knowledge Servers, IEEE Expert, Vol. 11, No. 3, 1996
59. Finin T., Labrou Y., Mayfield J.: KQML as an Agent Communication Language. Software Agents, AAAI Press, 1997, pp. 2291-316
60. Fujita S, et al.: OSEC Projects Description, in: Open Architecture Control Systems, ITIA Series, Vol. 2, 1998, pp. 135-157
61. Futó Iván (ed.): Mesterséges Intelligencia, Aula Kiadó, 1999
62. Genesereth M. R., R. E. Fikes (Editors): Knowledge Interchange Format, Version 3.0 Reference Manual. Computer Science Department, Stanford University, Technical Report Logic-92-1, June 1992.
63. GE Fanuc web oldal, CNC & Laser, 2001, http://www.gefanuceur.com/3/31.asp
64. Gensym ADE, Gensym User Society Annual Meeting, 1998, Newport, RA, USA
65. Gensym: A Strategic Choice for Improving Business Operations: G2 Classic, 2000 http://www.gensym.com/products/g2.htm
Gensym: G2 Ver. 5.0, Core Manuals, 1997: (1) G2 Class Reference Manual, (2) G2 Developer's Manual, (3) G2 Reference Manual, (4) G2 System Procedures Reference Manual, (5) G2 Gateway Bridge Developer's Guide
66. Gensym: G2-Corbalink User's Guide, 2000
67. Gilsinn, J., Rippey, W., Falco, J., Quinn, T., Russell, R., Stouffer, K.: A Welding Cell That Supports Remote Collaboration,: Proc. of the Ninth International Conference on Computer Technology in Welding, Detroit, MI, September 28 - 30, 1999
68. GNOSIS: Knowledge Systematization: Configuration Systems for Design and Manufacturing, Final Report of IMS Test Case 7, 1994
http://ksi.cpsc.ucalgary.ca/IMS/GNOSIS/
69. Green S, L. Hurst, B. Nangle, P. Cunningham, F. Somers, R. Evans: Software Agents: A review, Report of Trinity College Dublin, 1997
http://www.cs.tcd.ie/research_groups/aig/iag/toplevel2.html
70. Gruber T.R.: The Role of Common Ontology in Achieving Sharable, Reusable Knowledge Bases, Proc. of 2nd Int. Conf.: Principles of Knowledge Representation and Reasoning, Cambridge, MA, pp: 601-602, Morgan Kaufmann, 1991
71. Guh R.S., J.D.T. Tanock, C. O'Brien: IntelliSPC: a hybrid intelligent tool for on-line economical statistical process control, Expert Systems with App. 17 (1999) 195-212

72. Gullander P, A. Adlemo, S. Andréasson, M. Fabian, B. Lennartson: Guidelines for Generic Message-Based Structure to obtain Flexible Cell Control, Opening Productive Partnerships: Proceedings of the Conference on Integration in Manufacturing, Vienna, 1995, IOS Press
73. Haidegger G., R. Kuba: Highly automated flexible assembly system, Proc. of Automation'95 Conference, 1995. Sept. 5-7, Budapest, pp. 347-353
74. Hatvany J: The efficient use of deficient information, CIRP Annals, Vol 32/1 1983, pp. 423-425
75. Hernández J.Z., J.M. Serrano: Reflective knowledge models to support an advanced HCI for decision management, Expert Systems with Applications 19 (2000) 289-304
76. HIPARMS: Highly Productive and Reconfigurable Manufacturing Systems, abstract of IMS proposal, 1996, http://www.ims.org/projects/project_info/hiparms.html
77. Holonic Manufacturing Systems, http://hms.ifw.uni-hannover.de/
78. Horváth M, A. Márkus, J. Váncza: Co-operation via conflicts in manufacturing systems, in: Cooperative Knowledge Processing for Engineering Design, Kluwer, 1999 pp. 307-324
79. Horváth M, Somló J: A forgácsolási folyamatok optimálása és adaptív irányítása, Muszaki Könyvkiadó, Bydapest, 1979
80. Hung C.Y., R.T. Sumichrast: A multi-expert system for material cutting plan generation, Expert Systems with Applications 19 (2000) 19-29
81. Ibargüengoytia P.H, L.E. Sucar, S. Vadera: An expert system for sensor validation, Proc. of 4th World Congress on Expert Systems, Mexico City, March 16-20, 1998, pp. 106-113
82. IF7: Research on Innovative and Intelligent Field Factory (IF7), IMS project, 1996 http://www.ims.org/projects/project_info/if7.html
83. IMS Feasibility Study, Final Report of the International Steering Committee adopted at ISC6, Hawaii, 24 to 26 January, 1994 http://ksi.cpsc.ucalgary.ca/IMS/ISC/ISC94_Contents.html
84. IMS program honlapja, http://www.ims.org/
85. Intelligent Manufacturing report: Ford Installs Intelligent Cell Controller,Vol. 2, No. 7, 1996 July, http://lionhrtpub.com/IM/IMsubs/IM-7-96/ford.html
86. INTERGLOBAL: Intelligent Interface for Modeling, Simulation and Optimization coupled with Workflow Management and Production Planning and Control Systems for Global Manufacturing, IMS project, 1998 http://www.ims.org/projects/project_info/interglobal.html
87. ISO/DIS 14640-21, International automation systems and integration - Physical device control - Data modell for Computerized Numerical Controllers, Part 21, Mapping to STEP (ISO10303) Integrated Resources, Draft Version 1, 1999
88. Iwata K, M. Onosato, K. Teramoto, S. Osaki: Virtual Manufacturing Systems as Advanced Information Infrastructure for Integrating Manufacturing Resources and Activities, Annals of the CIRP, Vol. 46/1, 1997, pp. 335-338
89. JOP: Japan FA Open Systems Promotion Group, 1999, http://www.mstc.or.jp/jop/
90. JOP: PAPI, CNC Application Programming Interface, PAPI Specification 1.01E, 1999,
91. JOP: FL-net Protocol Specification Ver 1.0, 1999
92. Kaula R.: Communication Model for Module-Based Knowledge Systems, in. Knowledge Based Systems, Academic Press, Vol. 1, 2000
93. Kádár, B. S. Markos, L. Monostori: Knowledge Based Reactive Management of Manufacturing CellsConf. of Integration of Manufacturing, Galway, Irland, 2-4 Oct. 1996, pp. 197-205
94. Knowledge Technologies International: KBO Environment, 2001 http://www.ktiworld.com/our_products/index.shtml

95. Knutilla A., C. Schlenoff, R. Ivester: "A Robust Ontology for Manufacturing Systems Integration," Proceedings of 2nd International Conference on Engineering Design and Automation, Maui, Hawaii, August 7-14, 1998.
96. Koren Y, Z.J. Pasek, G. Ulsoy: Real-Time Open Control Architecture ControllersReconfigurable Manufacturing Systems, Keynote Paper, Annals of the CIRP Vol. 48/2/1999, pp. 527-540
97. Koren Y, U. Heisel, F. Jovane, T. Moriwaki, G. Pritschow, G. Ulsoy, H. Van Brussel: Reconfigurable Manufacturing Systems, Keynote Paper, Annals of the CIRP Vol. 48/2/1999, pp. 527-540
98. KUKA Roboter Gmbh, http://www.kuka-roboter.de/webc/re_engl/index.html
99. Kumara S.T, Kashyap R.L, Soyster A.L. (ed.): Artificial Intelligence, Manufacturing Theory and Practice, Inst. of Industrial Engineering, 1988
100. Kusiak A (ed.): Artificial Intelligence, Implications for CIM, IFS and Springer, 1988
101. Lee, K.I, S.D. Noh: Virtual Manufacturing System - a Test-Bed of Engineering Activities, Annals of the CIRP, Vol. 46/1, 1997, pp. 347-350
102. Lenat, D: CYC: A Large-Scale Investment in Knowledge Infrastructure, Comm. of ACM, 38(11) pp. 33-38, 1995; + Cyc Knowledge Server: http://www.cyc.com/products.html
103. Liang M.: An intelligent control system for CNC machining process, http://www.uottawa.ca/services/research/transfer/industry/152.htm
104. Lin A. D, Starr B.H: HIKE (HPKB Integrated Knowledge Environment), 1998 http://hike.saic.com/saic/documents/KDEX98/HPKB-KDEX98.htm
105. Lin Y. T., C. Zhou: Modeling and Analysis of Message Passing in Distributed Manufacturing Systems, IEEE Transactions on Systems, Man, and Cybernetics, vol. 29, no. 2, May 1999, pp. 250-262.
106. Liu Y., L. Zuo, T. Cheng, C. Wang: Development of an Open Parallel Intelligent CNC Milling System, Part I and II, Int. J. Adv. Manuf. Technol. (2000) 16:537-541, 542-446
107. Lutz, P: Comparison between the OSACA and OMAC API approaches on an Open Controller Architecture, in: Open Architecture Control Systems, ITIA Series, Vol. 2, 1998, pp.203-208
108. Manufacturing Data Systems: OpenCNC® Architecture és OpenCNC® Application Programming Interface, 2000, http://www.mdsi2.com
109. Márkus A, T. Kis, J. Váncza, L. Monostori: A market approach to holonic manufacturing, CIRP Annals, Vol. 45/1 pp. 433-436
110. Maropoulos, P.G.: Aggregate Product and Process Modelling for the Welding of Complex Fabrications, Annals of the CIRP Vol. 48/1/1999 pp. 401-404
111. Mazak Corp.: Mazatrol Fusion 640 CNC/PC Control, 2000, http://www.mazak.com
112. McGuire J. G., D. R. Kuokka, J. C. Weber, J. M. Tenenbaum, T. R. Gruber, G. R. Olsen.: SHADE: Technology for Knowledge-Based Collaborative Engineering, Journal of Concurrent Engineering: Applications and Research (CERA), 1(2), September 1993.
113. Mikosch Falk (ed.): Interoperability of Standards for Robotics in CIME, Research Reports Esprit, Vol. 1, Springer, 1996
114. Ming L, Y. Xiaohong, M. Tseng, Y. Shuzi: A CORBA-based agent-driven design for distributed intelligent manufacturing systems, J. of Intelligent Manufacturing (1998) 9, pp. 457-465
115. MISSION: Modelling and Simulation Environments for Design, Planning and Operation of Globally Distributed Enterprises, abstract of IMS project, 1996 http://www.ims-mission.de/
116. Mitsushi M, T. Nagao, H. Okabe, M. Hashiguchi, K. Tanaka: Annals of the CIRP, Vol 46/1, 1997, pp. 269-274

117. Monostori L.: B-spline option of the DIALOG CNC for turning, Internal Report, SzTAKI, Budapest, (in Hungarian), 1981
118. Monostori L: Intelligent Machines, Proc. of 2nd Conf. On Mechanical Engineering, Budapest, Hungary, May. 25-26, 2000, pp. 24-36
119. Monostori, L.; Kádár, B.: Holonic control of manufacturing systems, Preprints of the 1st IFAC Workshop on Multi-Agent-Systems in Production, December 2-4, 1999, Vienna, Austria, pp. 109-114.
120. Monostori L, A. Márkus, H. Van Brussel, E. Westkamper: Machine Learning approaches to manufacturing, CIRP Annals, Vol. 45, No. 2, pp. 675-712
121. Monostori L, Zs. J. Viharos, S. Markos: Satisfying varoius requirements in different levels and stages of machining using one general ANN-based process model, J. of Materials Processing Technology, Elsevire, 107 (1-3), 2000, pp. 228-235
122. Moon C. (ed.): Dynamics and Chaos in Manufacturing Processes, Wiley, 1998
123. Motion Engineering: XMP Series Product Overview, 1998, http://www.motioneng.com
124. Moriwaki T: Intelligent Machine Tool, Journal of JSME, Vol. 96, 1993, pp. 1010-1014
125. Nagy G., Haidegger G.: Distributed Control of Assembly Cells based on Virtual Manufacturing Device Modell", Proc. of IFAC Workshop on DCCS, Toledo, Spain, Sept. 1994.
126. NC Technika: NCT2000-es vezérlocsalád, 2000, http://www.nct.hu/NCT_termekek/Vezerlok/Nct2000/nct2000.html
127. NGMS – Next Generation Manufacturing Systems, Merging the Agile, Autonomous and Distributed, Biological, and Fractual Manufacturing Concepts, 1994 http://www.cam-i.org/ngms.html
128. NIIIP Consortium: NIIIP Reference Architecture, 1998 http://www.niiip.org/public/public-forum.html
129. Ohashi K., Shin K.: Model-based Control for Reconfigurable Manufacturing Systems, IEEE Int. Conf. on Robotics and Automation, Seoul, May 21-26, 2001
130. OMAC API Work Group: OMAC API SET, Version 0.23, Working Document, 1999 http://www.isd.mel.nist.gov/projects/omacapi/
131. OPTIMAL: New Programming Interface for CAM-CNC coupling, ESPRIT III. project, 1997, http://dbs.cordis.lu/
132. OSACA/HÜMNOS: Style Guide Werkzeugmaschinen, Fraunhofer IRB Verlag, 1997,
133. OSE Consortium, OSEC: Open System Env. for Controller Architecture Draft, 1996 http://www.mli.co.jp/OSE/,
134. OSE Consortium: Development of OSEC, OSEC-II Project Technical Report, 1998
135. Ott E., C. Grebogi, J. Yorke: Controlling Chaos, Phys. Rev. Lett. Vol. 64, 1990, pp. 1196-1199
136. Pack R.T., D. M. Wilkes, G. Biswas, and K. Kawamura: Intelligent Machine Architecture for Object-Based System Integration, Proc. of the 1997 IEEE/ASME Int. Conf. on Advanced Intelligent Mechatronics, Waseda University, Japan, June 1997.
137. Parunak, H. V. D.: A Practitioners's Review of Industrial Agent Applications, Autonomous Agents and Multi-Agent Systems 3:4, 2000
138. Patriti V, P. Charpentier: Application des SMA a la Fabrication Cas de Shiva, 3éme J. Francophones sur l'Intelligence Artificielle Distribuée et les Systémes Multi-Agents, St Bandolf, France, 1995
139. Peterson J.L.: Petri Nets Theory and the Modelling of Systems, Prentice-Hall, 1981
140. Petit M, E. Dubois: Defining an Ontology for the Formal Requirements Engineering of Manufacturing Systems, In Interprise Engineering and Integration, Proc. of International Conference on Enterprise Integration an Modeling Technology, Springer, 1997.

141. Pham D.T, H.L. Xing: Neural Networks for Identification, Prediction and Control, Springer, 1995
142. Predator Sotware Inc., Predator Editor, 2001, http://www.predator-software.com/editor.htm
143. Pritchow G, Ch. Daniel, G. Junghans, W. Sperling: Open System Controllers - A Challenge for the Future of the Machine Tool Industry, Annals of the CIRP, 42/1, 1993, pp. 449-452
144. Rao M., X. Sun, J. Feng: Intelligent system architecture for process support, Expert Systems with Applications 19 (2000) 279-288
145. Rippey, W.G., Falco, J.A: The NIST Automated Arc Welding Testbed, Proc. of the 7th Int. Conf. on Computer Technology in Welding, San Francisco, CA, July 8-11, 1997
146. Russel S.J, P. Norvig: Mesterséges Intelligencia modern megközelítésben, Panem - Prentice Hall, 2000
147. Rzevsky, G: Mechatronics: Designing Intelligent Machines, Butterworth, 1995
148. Sándor T.: PC control of desktop CNC milling machines, XV. Inter. Wissenschaftliches Kolloquim, 18-21 Apr. 1999, Bremen, Germany, Hochschule Bremen, pp. 195-201
149. Schneider G.: Control technology for industrial robots, Mastering diversity in the periphery by means of PC technology, KUKA Roboter GmbH, 1998, http://www.kuka-roboter.de
150. Schuszter Gy.: Fuzzy control of simulated mobile robot, XV. Inter. Wissenschaftliches Kolloquim, 18-21 Apr. 1999, Bremen, Germany, Hochschule Bremen, pp. 181-186
151. SEMATECH: Computer Integrated Manufacturing (CIM) Framework Specification, Version 2.0, 1998, http://www.sematech.org/public/resources/stds/cim_frmw.htm
152. SERCOS N.A.: Comparison of Technologies for Communication Between Digital Drives and Controls,2000, http://www.sercos.com/comparisons.htm
153. Shackleford W, F.M. Proctor: The Real-Time Control System Library, http://isd.cme.nist.gov/proj/rcs_lib
154. Shen, W., Norrie, D.H.: Agent-Based Systems for Intelligent Manufacturing: A State-of-the-Art Survey. Knowledge and Information Systems, an International Journal, 1(2), 129-156, 1999
155. Siemens: Automation & Drives, Catalog CD (CA 01, 10/99 kiadás), 1999
156. de Silva C.W: Intelligent Control, Fuzzy Logic Applications, CRC Press, 1995
157. SIMAC: Design System for Intelligent Machining Centres and Cells, abstract of IMS proposal, 2000, http://www.ims.org/projects/project_info/simac.html
158. SIMON: Sensor Fused Intelligent Monitoring System for Machining, abstract of IMS project, 1997, (Az ESPRIT SIMON projekten alapul az IMS javaslat is) http://www-wm.wzl.rwth-aachen.de/SIMON/index.html vagy http://www.ims.org/projects/project_info/simon.html
159. SISCO products: MMS-EASE, 2000, http://www.sisconet.com/mmsease.htm
160. Sopronfalvi Z., Damm Z., Káposztás P., Farkas R.: Helyi Irányító Rendszer, HIR, Rendszer Könyvek, Eurocom Rt., 1998
161. Sousa P, C. Ramos: A dynamic scheduling holon for manufacturing orders, J. of Intelligent Manufacturing (1998) 9, pp 107-112
162. STEP-Compliant Data Interface for Numeric Controls STEP-NC, abstract of IMS project, 1997, http://www.step-nc.org/
163. Su K.W., S.L. Hwang, T.H. Liu: Knowledge architecture and framework design for preventing human error in maintenance tasks, Expert Systems with Applications 19 (2000) 219-228
164. Syan C., Y. Mostefai: Status monitoring and error recovery in flexible manufacturing systems, Int. J. of Integ. Manuf. Systems, 6(4), 43-48, 1995

165. Szelke E., R.M. Kerr, Knowledge Based Reactive Scheduling - State of the Art, Int. J. of Production Planning and Control, 1994, Vol. 5, pp 124-145
166. Tanaya I.P., J. Detand, JP. Kruth: Holonic machine controller: A study and implementation of holonic behaviour to current NC controller, Computers in Industry 33 (1997) pp. 323-333.
167. Tecnomatix Inc.: Workplace Design Applications (formerly ROBCAD), 2001, http://www.tecnomatix.com/showpage.asp?page=211
168. Teti R, S.R.T. Kumara: Intelligent Computing Methods for Manufacturing Systems, Annals of Cirp, Vol 46/2, 1997, pp. 629-652
169. Tomiyama, T., Xue, D., Umeda, Y., Takeda, H., Kiriyama, T., and Yoshikawa, H., Systematizing Design Knowledge for Intelligent CAD Systems, Human Aspects in Computer Integrated Manufacturing, IFIP Transactions B-3, North- Holland 1992, pp. 237 - 248.
170. Tonkay G.L, K. Knott: An Expert System for Welding, in Artificial Intelligence, Manufacturing Theory and Practice (ed. Kumara, Kashyap, Soyster), Inst. of Industrial Engineering, 1988
171. Valckenaers P., H. Van Brussel: IMS TC5: Holonic Manufacturing Systems Technical Overview, 1994, http://icon.ncms.org/hms/public/hms_tech.html
172. Van Brussel H., J. Wyns, P. Valckenaers, L. Bongaerts, P. Peeters, Reference architecture for holonic manufacturing systems: PROSA, Computers in Industry 37 (3) (1998) pp. 255-274.
173. Váncza J., Horváth M., Stankóczi Z.: Robotic Inspection Plan Optimization by Case-Based Reasoning, Proc. of the 2nd World Congress on Intelligent Manufacturing Systems and Processes, June 1997, Budapest, Hungary, pp. 509-515
174. Veridian (1999): Expert System for Crash Data Collection, http://www.calspan.com/expert.html
175. Weiss G. (ed.): Multiagent Systems, A Modern Approach to Distributed Artificial Intelligence, MIT Press, 1999
176. Wright P.K, D.A. Bourne: Manufacturing Intelligence, Addison-Wesley, 1988
177. Yamazaki K, Y. Hanaki, M. Mori, K. Tezuka: Autonomously Proficient CNC Controller for High-Performance Machine Tools Based on an Open Architecture Concept, Annals of the CIRP, Vol. 46/1, 1997, pp. 275-278
178. Zhang L., J. Deng, S.C.F. Chan: A Next Generation NC Machining System Based on NC Feature Unit and Real-Time Tool-Path Generation, Int. J. Adv. Manuf. Technol. (2000) 16: 889-901
179. Zhang, X., Norrie, D.H.: Holonic Control at the Production and Controller Levels", Proc. of the 2nd International Workshop on Intelligent Manufacturing Systems, Leuven, Belgium, pp. 215-224, September 22-24, 1999.
180. Zhou, B., Wang, L., Norrie, D.H., " Design of Distributed Real-Time Control Agents for Intelligent Manufacturing Systems", Proc. of the 2nd International Workshop on Intelligent Manufacturing Systems, Leuven, Belgium, pp. 237-244, September 22-24, 1999.

11 FÜGGELÉK

11.1 AZ OSACA AC MODUL API SPECIFIKÁCIÓJA

változó név	**jelentés**
ac_command_interface_i - 1	aktív adat megadása
ac_command_interface_i - 2	tengely konfigurációja
ac_current_axes_values_i	aktuális tengelyérték lekérdezése
ac_axis_display_data	a tengely értékek együttes lekérdezése
ac_current_position_acs_k	a tengely aktuális pozíciójának lekérdezése
ac_active_position_acs_k	a tengely setpoint pozíciójának lekérdezése
ac_measurement_value_k	a pillanatnyi mért tengely érték
ac_status_k	tengely státusz információ
ac_release_feed_k	feed engedélyezése/tiltása
ac_follow_up_operation_k	tracking engedélyezése/tiltása
ac_controller_enable_k	tengely mozgatás engedélyezése/tiltása

11.2 AZ OSEC SZERVO API SPECIFIKÁCIÓJA

függvény név	**jelentés**
OsecSrvOpen()	szervo interfész megnyitása
OsecSrvClose()	szervo interfész lezárása
OsecSrvInit()	szervo interfész inicializálása
OsecSrvSetInfo()	szervo interfész info beállítása
OsecSrvGetInfo()	szervo interfész info lekérdezése
OsecSrvGrpInit()	szervo csoport inicializálása
OsecSrvGrpSetInfo()	szervo csoport info beállítása
OsecSrvGrpGetInfo()	szervo csoport info lekérdezése
OsecSrvAxisInit()	szervo tengely inicializálása
OsecSrvAxisSetInfo()	szervo tengely info beállítása
OsecSrvAxisGetInfo()	szervo tengely info lekérdezése
OsecSrvGetFreeBufCount()	szabad buffer mérete
OsecSrvGetUsedBufCount()	foglalt buffer mérete
OsecSrvCancelBuf()	buffer érvénytelenítése
OsecSrvErrorReset()	szervo hibajelek alapállapotba
OsecSrvSetSrvOn()	szervo be
OsecSrvSetSrvOff()	servo ki
OsecSrvSetCyclicCmd()	ciklikus parancsmegadás beállítása
OsecSrvGetCyclicData()	ciklikus adat lekérdezés
OsecSrvSyncroCyclicData()	ciklikus adat szinkronizálás
OsecSrvWaitEvent()	szevo eseményre várakozás

11.3 AZ OMAC VEZÉRLO TENGELY (AXIS) SPECIFIKÁCIÓJA

Az OMAC 'axis' modul API specifikációja, ami több interfészbol épül fel (alap/Iaxis, input/output parancs, engedélyezés/leállítás, korlátok/méretek, pozíció / sebesség / áram szervo, homing/jogging, inkrementális, abszolút pozícionálás, alapállapotba helyezés, setup, felügyelet, érzékelés). A "!" jelzi, hogy a minimális készletbe tartozik a metódus.

interfész név		eljárás név
IAxis	!	currentStateName(BSTR * name),
	!	disableAxis(),
	!	enableAxis(),
		endCommandedPosition(),
		endCommandedTorque(),
		endCommandedVelocity(),
	!	estopAxis(),
	!	followCommandedAbsPosition(),
	!	followCommandedRelPosition(),
	!	followCommandedTorque(),
	!	followCommandedVelocity(),
	!	getCommandedInput(IAxisCommandedInput ** ppIAxisObject),
	!	getCommandedOutput(IAxisCommandedOutput** ppIAxisObject),
		getDisabling(IAxisDisabling ** ppIAxisObject),
		getDynamics(IAxisDyn ** ppIAxisObject),
		getEnabling(IAxisEnabling ** ppIAxisObject),
		getHomeOffset(double * pOffset),
	!	getKinematics(IAxisKinematics ** ppIAxisObject),
	!	getLimits(IAxisLimits ** ppIAxisObject),
		getMaintenance(IAxisMaintenance ** ppIAxisObject),
	!	getPositionControlLaw(IControlLaw ** ppIControlLawObject),
	!	getPositioningServo(IAxisPositioningServo ** ppIAxisObject),
		getResetting(IAxisResetting ** ppIAxisObject),
	!	getSensedState(IAxisSensedState ** ppIAxisObject),
		getSetup(IAxisSetup ** ppIAxisObject),
	!	getTorqueControlLaw(IControlLaw ** ppIControlLawObject),
	!	getTorqueServo(IAxisTorqueServo **ppIAxisObject),
	!	getVelocityControlLaw(IControlLaw ** ppIControlLawObject),
	!	getVelocityServo(IAxisVelocityServo ** ppIAxisObject),
	!	holdCommandedPosition(),
	!	holdCommandedTorque(),
	!	holdCommandedVelocity(),
	!	homed(),
	!	isDisabled(boolean * b),
		isDisabling(boolean * b),
	!	isEnabled(boolean * b),
		isEnabling(boolean * b),
	!	isEstopped(boolean * b),
		isEstopping(boolean * b),
	!	isFaulted(boolean * b),
	!	isFollowingPosition(boolean * b),
	!	isFollowingTorque(boolean * b),
	!	isFollowingVelocity(boolean * b),
	!	isHoldingPosition(boolean * b),

! isHoldingTorque(boolean * b),
! isHoldingVelocity(boolean * b),
! isHomed(boolean * b),
! isReady(boolean * b),
isResetting(boolean * b),
loseHome(),
! processServoLoop(),
! resetAxis(),
! setCommandedInput(IAxisCommandedInput * val),
! setCommandedOutput(IAxisCommandedOutput * val),
setDisabling(IAxisDisabling * val),
setEnabling(IAxisEnabling * val),
setHomeOffset(double offset),
setKinematics(IAxisKinematics * val),
setLimits(IAxisLimits * val),
setMaintenance(IAxisMaintenance * val),
! setPositionControlLaw(IControlLaw * val),
! setPositioningServo(IAxisPositioningServo * val),
setResetting(IAxisResetting * val),
! setSensedState(IAxisSensedState * val),
setSetup(IAxisSetup * val),
! setTorqueControlLaw(IControlLaw * val),
! setTorqueServo(IAxisTorqueServo * val),
! setVelocityControlLaw(IControlLaw * val),
! setVelocityServo(IAxisVelocityServo * val),
! stopAxis()

IaxisCommandedInput

! getAbsolutePositioning(boolean * pBoolean),
! getAccelerationCmdInput(AxisAccelCmd * pVal),
! getForceCmdInput(AxisForceCmd * pVal),
! getPositionCmdInput(AxisPositionCmd * pVal),
! getRelativePositioning(boolean * pBoolean),
! getVelocityCmdInput(AxisVelocityCmd * pVal),
! setAbsolutePositioning(boolean val),
! setAccelerationCmdInput(AxisAccelCmd accelerationCmd),
! setForceCmdInput(AxisForceCmd forceCmd),
! setPositionCmdInput(AxisPositionCmd positioningCmd),
! setRelativePositioning(boolean val),
! setVelocityCmdInput(AxisVelocityCmd velocityCmd),
! updateCommandedInput()

IaxisCommandedOutput

! getForceCmdOutput(AxisForceCmd * pVal),
! getPositionCmdOutput(AxisPositionCmd * pVal),
! getTorqueCmdOutput(AxisTorqueCmd * pVal),
! getVelocityCmdOutput(AxisVelocityCmd * pVal),
! setForceCmdOutput(Force forceCmd),
! setPositionCmdOutput(Length positioningCmd),
! setTorqueCmdOutput(AxisTorqueCmd accelerationCmd),
! setVelocityCmdOutput(Velocity velocityCmd),
! updateCommandedOutput()

IAxisDisabling isDisabling::isDisabling(boolean * b),
isFailed(boolean * b),
startDisablingAxisAction(),
updateDisabledAxisAction(),
updateDisablingAxisAction()

IAxisEnabling isEnabling::isEnabling(boolean * b),
startEnablingAxisAction(),
updateEnabledAxisAction(),
updateEnablingAxisAction()

IAxisLimits getCutOffPosition(Length *pVal),
getFollowingErrorViolationLim(Length *pVal),
getFollowingErrorWarnLim(Length *pVal),
getHardFwdOTravelLim(Length *pVal),
getHardRevOTravelLim(Length *pVal),
getJerkLimit(Jerk *pVal),
getMaxForceLimit(Force *pVal),
getMaxVelocity(Velocity *pVal),
getOvershootViolationLim(Length *pVal),
getOvershootWarnLevelLimit(Length *pVal),
getSoftFwdOTravelLim(Length *pVal),
getSoftRevOTravelLim(Length *pVal),
getUndershootViolationLim(Length *pVal),
getUndershootWarnLevelLimit(Length *pVal),
getUsefulTravel(Length *pVal),
setCutOffPosition(Length newVal),
setFollowingErrorViolationLim(Length newVal),
setFollowingErrorWarnLim(Length newVal),
setHardFwdOTravelLim(Length newVal),
setHardRevOTravelLim(Length newVal),
setJerkLimit(Jerk newVal),
setMaxForceLimit(Force newVal),
setMaxVelocity(Velocity newVal),
setOvershootViolationLim(Length newVal),
setOvershootWarnLevelLimit(Length newVal),
setSoftFwdOTravelLim(Length newVal),
setSoftRevOTravelLim(Length newVal),
setUndershootViolationLim(Length newVal),
setUndershootWarnLevelLimit(Length newVal),
setUsefulTravel(Length newVal)

IaxisPositioningServo
endFollowingPositionAction(),
estopFollowingPositionAction(),
holdFollowingPositionAction(),
! isFollowingPositionError(boolean * b),
isStoppingFollowingPosition(boolean * b),
! resetFollowingPositionAction(),
! startFollowingAbsPositionAction(),
! startFollowingRelPositionAction(),
! stopFollowingPositionAction(),
updateEstoppingFollowingPositionAction(),
! updateFollowingPositionAction(),
updateHoldingFollowingPositionAction(),
updateResettingFollowingPositionAction(),
updateStoppingFollowingPositionAction()

IAxisResetting isResetting::isReset(boolean * b),
isResetting::isResetting(boolean * b),
startResettingAxisAction(),
updateResetAxisAction(),
updateResettingAxisAction()

IAxisSetup getCurrentRates(IAxisRates **pVal),
getDynamicRates(IAxisDyn **pVal),
getPhysicalLimits(IAxisRates **pVal),
setCurrentRates(IAxisRates * newVal),
setDynamicRates(IAxisDyn * newVal),
setPhysicalLimits(IAxisRates * newVal)

IaxisSensedState
! getActualAcceleration(AxisAccelCmd * a),
! getActualForce(AxisForceCmd * a),
! getActualPosition(AxisPositionCmd * a),
! getActualVelocity(AxisVelocityCmd * a),
getEnablingPrecondition(boolean * b),
! inPosition(boolean * pVal),
isEnablingPrecondition(boolean * pVal),
isFollowingErrorViolation(boolean * pVal),
isFollowingErrorWarn(boolean * pVal),
isHardFwdOTravel(boolean *pVal),
isHardRevOTravel(boolean * pVal),
isOverShootViolation(boolean * pVal),
isSoftFwdOTravel(boolean *pVal),
isSoftRevOTravel(boolean *pVal)

IaxisTorqueServo
endFollowingTorqueAction(),
estopFollowingTorqueAction(),
holdFollowingTorqueAction(),
! isFollowingTorqueError(boolean * b),
isStoppingFollowingTorque(boolean * b),
! resetFollowingTorqueAction(),
! startFollowingTorqueAction(),
! stopFollowingTorqueAction(),
updateEstoppingFollowingTorqueAction(),
! updateFollowingTorqueAction(),
updateHoldingFollowingTorqueAction(),
updateResettingFollowingTorqueAction(),
updateStoppingFollowingTorqueAction()

IaxisVelocityServo
endFollowingVelocityAction(),
estopFollowingVelocityAction(),
holdFollowingVelocityAction(),
! isFollowingVelocityError(boolean * b),
isStoppingFollowingVelocity(boolean * b),
! resetFollowingVelocityAction(),
! startFollowingVelocityAction(),
! stopFollowingVelocityAction(),
updateEstoppingFollowingVelocityAction(),
! updateFollowingVelocityAction(),
updateHoldingFollowingVelocityAction(),
updateResettingFollowingVelocityAction(),
updateStoppingFollowingVelocityAction()

IAxisSupervisor getAbsolutePos(IAxisAbsolutePos ** ppIAxisObject),
getHoming(IAxisHoming ** ppIAxisObject),
getIncrementPosition(IAxisIncrementPos ** ppIAxisObject),
getJogging(IAxisJogging ** ppIAxisObject),
getServoAxis(IAxis ** ppIAxisObject),
home(double velocity),

isHoming(boolean * b),
isIncrementingPosition(boolean * b),
isJogging(boolean * b),
isMovingto(boolean * b),
jog(double velocity),
runInterpolator(),
setAbsolutePos(IAxisAbsolutePos * val),
setHoming(IAxisHoming * val),
setIncrementPosition(IAxisIncrementPos * val),
setJogging(IAxisJogging * val),
setServoAxis(IAxis * val),

IaxisAbsolutePositioning

! abnormalStopAbsolutePosAction(),
finishedAbsolutePosAction(),
hardStopAbsolutePosAction(),
isAbnormalStopped(boolean * b),
isAbnormalStopping(boolean * b),
isCompleted(boolean * b),
isFinished(boolean * b),
isHardStopped(boolean * b),
isHardStopping(boolean * b),
isNormalStopped(boolean * b),
isNormalStopping(boolean * b),
isStopped(boolean * b),
isStopping(boolean * b),
! resetAbsolutePosAction(),
! startAbsolutePosAction(double position),
! stopAbsolutePosAction(),
updateAbnormalStoppingAbsolutePosAction(),
! updateAbsolutePosAction(),
updateHardStoppingAbsolutePosAction(),
updateResettingAbsolutePosAction(),

IAxisHoming

! abnormalStopHomingAction(),
finishedHomingAction(),
hardStopHomingAction(),
! resetHomingAction(),
! startHomingAction(),
! stopHomingAction(),
updateAbnormalStoppingHomingAction(),
updateHardStoppingHomingAction(),
! updateHomingAction(),
updateResettingHomingAction(),
updateStoppingHomingAction()

IAxisJogging

! abnormalStopJoggingAction(),
finishedJoggingAction(),
hardStopJoggingAction(),
! resetJoggingAction(),
! startJoggingAction(double targetVelocity),
! stopJoggingAction(),
updateAbnormalStoppingJoggingAction(),
updateHardStoppingJoggingAction(),
! updateJoggingAction(),
updateResettingJoggingAction(),
updateStoppingJoggingAction()

IAxisIncrementing		
	!	abnormalStopIncrementingAction(),
		finishedIncrementingAction(),
		hardStopIncrementingAction(),
	!	resetIncrementingAction(),
	!	startIncrementingAction(double incrementalPosition),
	!	stopIncrementingAction(),
		updateAbnormalStoppingIncrementingAction(),
		updateHardStoppingIncrementingAction(),
	!	updateIncrementingAction(),
		updateResettingIncrementingAction(),
IAxisRates		getMaxAcceleration(Acceleration *pVal),
		getMaxJerk(Jerk *pVal),
		getMaxTravel(Length *pVal),
		getPosErrRatioCutMoving(Length *pVal),
		getPosErrRatioIdleMoving(Length *pVal),
		getPosErrRatioIdleStationary(Length *pVal),
		getRepeatability(long *pVal),
		setMaxAcceleration(Acceleration newVal),
		setMaxJerk(Jerk newVal),
		setMaxTravel(Length newVal),
		setPosErrRatioCutMoving(Length newVal),
		setPosErrRatioIdleMoving(Length newVal),
		setPosErrRatioIdleStationary(Length newVal),
		setRepeatability(long newVal)

11.4 AZ OMAC VEZÉRLÉS ALGORITMUS (CONTROL LAW) SPECIFIKÁCIÓJA

Az OMAC 'control law' modul API specifikációja, ami az adott tengely szabályozási algoritmusának ad keretet.

interfész név		**eljárás név**
IControlLaw	!	getActualOffset([out,retval] double * val);
	!	getActualPosition([out,retval] double * val);
	!	getFollowingError([out,retval] double * val);
		getFollowingErrorOffset([out,retval] double * val);
	!	getOutputCommand([out,retval] double * val);
	!	getOutputOffset([out,retval] double * val);
	!	getSetpoint([out,retval] double * val);
	!	getSetpointDot([out,retval] double * val);
	!	getSetpointDotDot([out,retval] double * val);
	!	setActualOffset([in] double k);
	!	setActualPosition([in] double x);
		setFollowingErrorOffset([in] double off) ;
	!	setOutputCommand([in] double value);
	!	setOutputOffset([in] double k);
	!	setSetpoint([in] double X);
	!	setSetpointDot([in] double Xdot);
	!	setSetpointDotDot([in] double Xdotdot);
	!	getCycleTime([out,retval] double * val);
	!	setCycleTime([in] double time);

```
                    ! calculateOutputCommand();
                      setTuneIn(double value);
                      getTuneIn(double * value);
                    ! breakLoop();
                    ! makeLoop();
IPIDControlLaw
                      getKaf([out,retval] double * val);
                      getKcf([out,retval] double * val);
                      getKd([out,retval] double * val);
                      getKi([out,retval] double * val);
                      getKp([out,retval] double * val);
                      getKvf([out,retval] double * val);
                      getIntegrationLimit([out,retval] double * val);
                      setKaf([in] double k) ;
                      setKcf([in] double k);
                      setKd([in] double k);
                      setKi([in] double k);
                      setKp([in] double k);
                      setKvf([in] double k);
                      setIntegrationLimit([in] double integrationLimit);
IcontrolLawModuleClassFactory
                      CreateModule(name, riid, retval, iid_is(riid)] void ** ppvObj);
```

11.5 STEP NYELVU ÍVHEGESZTO ROBOTPROGRAM

```
STEP;
HEADER;
FILE_NAME('EXAMPLE PROARC INTERFACE #1', '1995-02-13 T15:00:00', 'JANOS NACSA',
('MTA SZTAKI', 'BUDAPEST HUNGARY'),
'STEP VERSION 1.0'
'MANUAL TYPING'
'ISO CD 10303 - 21'
'APPROVED BY STEPHAN PEPER');
FILE_DESCRIPTION('A SIMPLE ROBOT MOVEMENT IN PROARC INTERFACE',
'NOTHING SPECIAL);
FILE_SCHEMA('PROCESS_SEQUENCE_SCHEMA');
ENDSEC;
DATA;
#60=!POINT_PARAMLIST(TRUE,#150,FALSE,$,34.0,3.5,5.6,FALSE,$,4.2);
#62=!POINT_PARAMLIST(FALSE,$,TRUE,#140,44.0,4.5,6.6,TRUE,32.0,3.2);
#70=!GAP_INFO(1,2.3,4);
#80=!POSE((1016.1,108.1,1013.1),.ORYZYZ.,(-165.925,-3.518,102.516));
#81=!POSE((814.1,57.6,1016.8),.ORYZYZ.,(-167.331,-1.283,98.359));
#82=!POSE((804.4,89.7,908.8),.ORYZYZ.,(-164.771,3.934,96.150));
#83=!POSE((948.4,131.6,900.9),.ORYZYZ.,(-163.061,0.321,99.344));
#85=!POSE((234.5,856.7,89.0),.ORYZYZ.,(53.5,-4.6,76.9));
#86=!POSE((5234.5,56.7,189.0),.ORYZYZ.,(3.5,-64.6,6.9));
#87=!POSE((524.5,516.7,19.0),.ORYZYZ.,(32.5,-4.6,66.9));
#90=!JOINT((98.7,87.6,76.5,65.4,54.3,43.2),(32.1,21.0));
#91=!JOINT((198.7,887.6,96.5,665.4,754.3,443.2),(532.1,221.0));
#92=!JOINT((298.7,787.6,76.5,565.4,854.3,343.2),(632.1,121.0));
#93=!JOINT((398.7,687.6,176.5,465.4,954.3,243.2),(732.1,21.0));
#95=!JOINT((98.7,587.6,76.5,365.4,4.3,143.2),(32.1,921.0));
#96=!JOINT((498.7,87.6,276.5,65.4,54.3,43.2),(832.1,21.0));
#97=!JOINT((598.7,857.6,256.5,65.4,55.3,45.2),(852.1,25.0));
#100=!MOVE(.ptp.,#80,#90,90.0);
#101=!MOVE(.ptp.,#81,#91,70.0);
#102=!MOVE(.line.,#82,#92,52.2);
#103=!MOVE(.line.,#83,#93,62.2);
#105=!MOVE(.line,#85,#95,82.2);
#106=!MOVE(.line.,#86,#96,92.2);
#107=!MOVE(.line.,#87,#97,100.0);
#110=!SEAM_PARAMLIST(1.1,2.2,3.4);
#120=!WELDING_POINT(#105,#60,FALSE,FALSE,$);
#121=!WELDING_POINT(#106,#62,TRUE,TRUE,#70);
#130=!WOBBLE(1.0,2.0,3.0,4.0,5.0);
#140=!FILL_PARAMLIST(24.0,2.5);
#150=!HOT_PARAMLIST(123.456);
#180=!POSITIONING(1,'go close to the  plate',.robot.,(#100,#101));
#181=!OFFLINE_SEARCH(2,'search the corner',#102,.MS.,(#103,#107));
#182=!WELDING(3,'a line welding',TRUE,#130,(#120,#121),#110);
#200=!PROCESS_SEQUENCE($,(#180,#181,#182));
ENDSEC;
ENDSTEP;
```

11.6 TENGELY TESZTELO INTERFÉSZE

```
// AiAxisTest.idl
// NJ

module AiAxisTest {

interface AxisTest
{
      void init();
      void setAxis(in string ior);
      void start(in short testnum);
      void stop();
      string infoMessage();
};

};


// KscTester.idl
// NJ

module KscTester {

interface KscAxisTester
{
      void init();
      void setAxis(in string ior);
      void start(in short testnum);
      void stop();
      string KscMessage();
      double getJogPos(in short num);
      double getVelo(in short num);
      boolean sendHome();
      void setActPos(in short num, in double pos);
};

};
```