Robotrendszerek Programozása

Moduláris Robotszoftver Keretrendszerek

Dr. Galambos Péter
-
Robottechnikai Szakkollégium

NTP-SZKOLL-20-0043 Robottechnikai Szakkollégium - Tehetséggondozás és szakmai közösségépítés az OE ROSZ-ban – 1 000 000 Ft támogatás

MINISZTERELNÖKSÉG

Bevezetés

A közelmúltban (2000-2015) megfigyelhető technológiai fejlődés számos ponton paradigmaváltást eredményez a robotrendszerek működési elvét érintő területeken. Ezen technológiák legfőképpen az alábbiak:

- **Infokommunikáció:** alacsony késleltetésű, szélessávú vezetékes és mobil hálózatok
- **Szoftver-interoperabilitás:** heterogén, elosztott, moduláris szoftverrendszerek
- **Felhő alapú számítástechnika:** IaaS, PaaS, SaaS
- **Valóssidejű térérzékelés:** RGB-D szenzorok, LiDAR, struktúrált fény elven működő szenzorok
- **Virtuális és kiterjesztett valóság:** Oculus VR, MS Hololens, stb.
- **Szemantikus technológiák:** Ontológiák, Fogalmi szintű következtetés
- **Gépi tanulás:** Együttes módszerek, mély neurális hálózat alapú tanulás
- **Akkumlátorok:** 3D nanostruktúra, új elektróda anyagok

Ez a fejlődés együttesen eredményezi napjaink látványos változásait a Robotika területén. A dinamikus változások egyik fő mozgatója a komponens-alapú szoftver környezetek megjelenése. Az előadás keretében ezek fejlődésével és alapvető paradigmáival ismerkedünk meg.

Komponens keretrendszerek – motiváció I.

Az ipari robottechnika gyökerei az 1960-as évekig nyúlnak vissza (Unimate, USA). Azóta a robotok szoftverei a gyártók által hagyományosan féltve őrzött szellemi termkek zárt forráskóddal. Ez egyfelől a know-how jelentős értékével, másrészt a robotok használatához kapcsolódó jogi, felelősségi kérdésekkel magyarázható.

Az ezredfordulóra a robottechnika terjedése és főként a kissebb vállalkozások piacra lépése, valamint az egyetemi oktatás új igényeket támasztott, amelyek révén megjelentek és népszerűvé váltak a kísérleti nyílt forrású robot szoftverek és programkönyvtárak.

Főként az egyetemi kutatólaboratóriumok tevékenysége nyomán kialakultak olyan keretrendszerek, amelyek egységes formában alapvető funkciókat biztosítanak a robot szoftverek készítéséhez, elfedve a fejlesztők elől az alacsony-szintű részfunkciók (pl. kommunikáció, konfiguráció és állapot menedzsment, hibakezelés, stb.) komplexitását. Az ilyen típusú szoftver környezetek elősegítik azt a folyamatot, amelynek eredmény képpen de-facto szabványok alakulnak ki az egyes robotikai funkciócsaládok interfészeire.

Ez a folyamat a fejlődésének kezdeti szakaszánál tart és számos vonatkozásban még kiforratlanok a rendelkezésre álló szoftver eszközök. Ez természetesnek tekinthető, mivel a kiterjedt kísérleti használat során fogalmazódnak meg azokaz igények, amelyek révén végül ipari szinten is elterjednek az új típusú moduláris robot-szoftverek.

Kapcsolódó források, érdekességek:

http://www.robothalloffame.org/inductees/03inductees/unimate.html

http://www.everything-robotic.com/2013/10/open-vs-closed-robot-systems.html

Komponens keretrendszerek – motiváció II.

A modularitásra és a komponens szintű interfészek szabványosítására kialakuló igényt egy egyszerű példával igyekszünk szemléltetni:

Látórendszer illesztése ipari robothoz

A hagyományos (proprietary) robotos környezben a látórendszer illesztését kizórlag a robot gyártója által támogatott hardver és szoftver eszközökkel lehetséges megoldani. Tipikusan szerződött szervizpartnerek és rendszer-integrátorok részvételével valósulhat meg egy ilyen beruházás. A verseny hiánya a részegységek (kamera, gépi látás szoftver, stb.) szintjén rendszerint igen magas árakat eredményez.

Moduláris megközelítésben mind a kamera, mind pedig szükséges szoftver kompenensek több forrásból beszerezhetőek a nyílt és egyezményes hardveres és szoftveres interfészeknek köszönhetően, vagy akár saját fejlesztésként is magvalósíthatóak.

Az interfészek szabványosítása az “okostelefonokon” megszokott alkalmazás áruházak mintájára a szervizrobotok terén lehetővé teszi a robotok képességrendszerének egyéni összeállítását. A fejlett, nagysebességű távközlőhálózatok és a felhő technológiák segítik ennek megvalósulását.

Komponens keretrendszerek – történelem

1. Robot Technology Middleware (RT-Middleware)
   - 2004, Noriaki Ando (AIST, Japan)
   - OMG RTC 1.1 Standard
   - Interoperabilitás alrendszer: CORBA (http://www.corba.org/)
   - A gyenge minőségű angol nyelvű dokumentáltság miatt elsősorban Japánban és a távolkeleten terjedt el

2. Microsoft Robotics Developer Studio (MRDS)
   - 2007, Microsoft
   - Nincsen szabványos háttere
   - Interoperabilitás alrendszer: DSSP (http://purl.org/msrs/dssp.pdf)
   - Nem terjedt el

3. Robot Operating System (ROS)
   - 2009, Willow Garage, USA
   - Nincsen szabványos háttere
   - Interoperabilitás alrendszer: XMLRPC (http://wiki.ros.org/ROS/Technical%20Overview)
   - Világszerte elterjedt.
   - A jelenlegi rendszer gyengeségei miatt új változat (ROS 2.0) fejlesztése folyik, amelyben az XMLRPC helyét a DDS (http://portals.omg.org/dds/) váltja fel.

Komponens keretrendszerek – Kommunikácó I.

A moduláris keretrendszerek tipikusan 2-féle adatátviteli módot tesznek lehetővé.

Data flow (adatfolyam): Az adatcsere egy egyszerű egyirányú “csővezeték” analógiával írható le. Az küldő és a fogadó fél között egy típusos adatcsatornában jutnak el az adatcsomagok a küldőtől a fogadó félig. A típusosság azt jelenti, hogy csak az előre definiált adattípus továbbítható a csatornán (pl. egyszerű típusok: int, float; vagy összetett típusok: RGBColor, vector3D, stb.). A komponensek összekötése az egyes rendszerekben másképp valósul meg. Az RT-Middleware egy pont-pont összeköttetésen alapuló modellt követ, amíg a ROS a publish-subscribe filozófián alapul. A felszínen különböző koncepciók ellenére az alacsonyszintű működés mechanizmusai azonos elven működnek. Mind a küldő, mind a fogadó félnek ismernie kell a csatorna adattípusát.

Remote Procedure Call (távoli eljáráshívás): Az adatcsere kétirányú és a függvényhívás analógia mentén írható le. A fogyasztó (consumer) használja a szolgáltató (provider) fél által felkínált szolgáltatásokat. A szolgáltatások definícióját interfésznek nevezzük. Ugyanazon interfésznek többféle implementációja lehetséges hasonlóan az objektum orientált programozásban megszokott mintához.

A midleware-ek (a háttérben) az adatfolyamot is az RPC mechanizmus útján valósítják meg. A Dataflow egy egyszerűsített absztrakciónak gogható fel, amely a távoli eljáráshívás speciális eseteként valósul meg. Mindegyik rendszer esetén elengedhetetlen az adattípusok és interfészek egyértelmű és nyelvfüggetlen definíciója. Az RT-Middleware esetében ez a leírás az IDL (Interface Definition Language) használatával történik, ami a CORBA szabványhoz köthető.

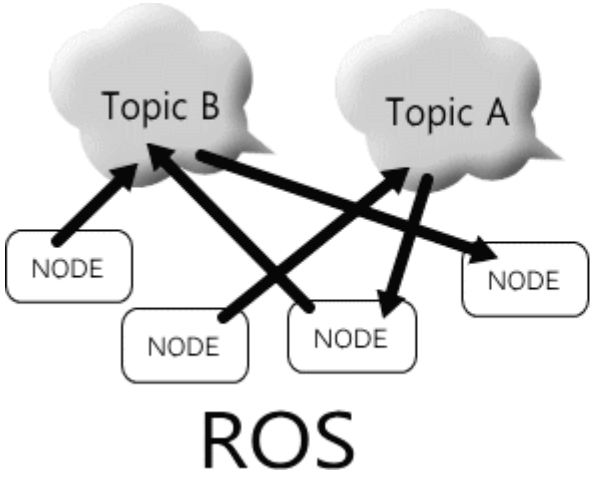
Komponens keretrendszerek – Kommunikácó II.

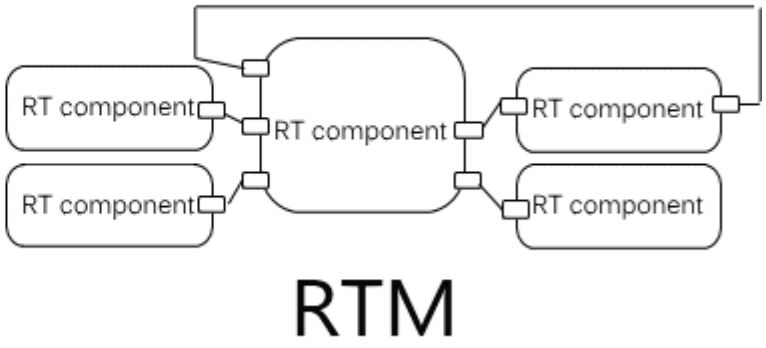
Adatfolyam (Dataflow) összeköttetés megvalósulása a ROS és RTM rendszerekben

ROS – Publish subscribe modell: A ROS keretrendszerben ún. topic-okat hozunk létre, amelyekre az egyes komponensek (node-ok) előfizetnek (subscribe), míg más komponensek a topic-okba üzeneteket küldenek (publish). Az üzenet (message) típusa a topic-hoz kötött.

RTM – pont-pont összeköttetés: Az RTM esetében a komponenseken ún. adatportokat hozunk létre, amelyeknek iránya lehet kimenet (out port), vagy bemenet (in port). Egy rendszer felépítésekor a megfelelő kapukat explicite összekötjük egy kapcsolat (connection) létrehozásával. Egy kimentre több bemenet csatlakoztatható.

Mivel a háttérben a ROS is egyedi kapcsolatokat tart nyilván és működtet TCP/IP szinten, ezért alacsonyszintű működést tekintve a kétféle megközelítés nem különbözik.





Forrás: Yaki Suga

Komponens keretrendszerek – Kommunikácó III.

Szolgáltatás (Service) – RPC típusú interakció

Az RTM-ben alkalmazott adatport és a ROS publish-subscribe data-flow rendszere nagyon rugalmas kommunikációs paradigma, de ezekkel nem valósítható meg az RPC típusú kérés-válasz jellegű kommunikáció, amely gyakran szükségszerű az elosztott rendszerekben. A kérés-válasz interakció szolgáltatásokon (service) valósul meg amelyhez két üzenet tartozik: egy a kéréshez (request) és egy a válaszhoz (reply).

RPC megvalósítása ROS környezetben
Egy szolgáltató (provider) ROS node névvel (string) azonosítja és ajánlja ki a szolgáltatást. A kliens, azaz a fogyasztó egy kérés üzenet küldésével hívja a szolgáltatást és vár a válaszra. A kliens programkönyvtárak olyan absztrakción keresztül valósítják meg az üzenetváltást, mintha tényleges eljáráshívás valósulna meg. A szolgáltatások leírása srv fileokon keresztül történik, amelynek szintaxisát a ROS definiálja.
A 2016-ban megjelenő ROS 2.0-ban a service-ek megvalósítása a DDS (Data Distribution Service) technológiával valósul meg.

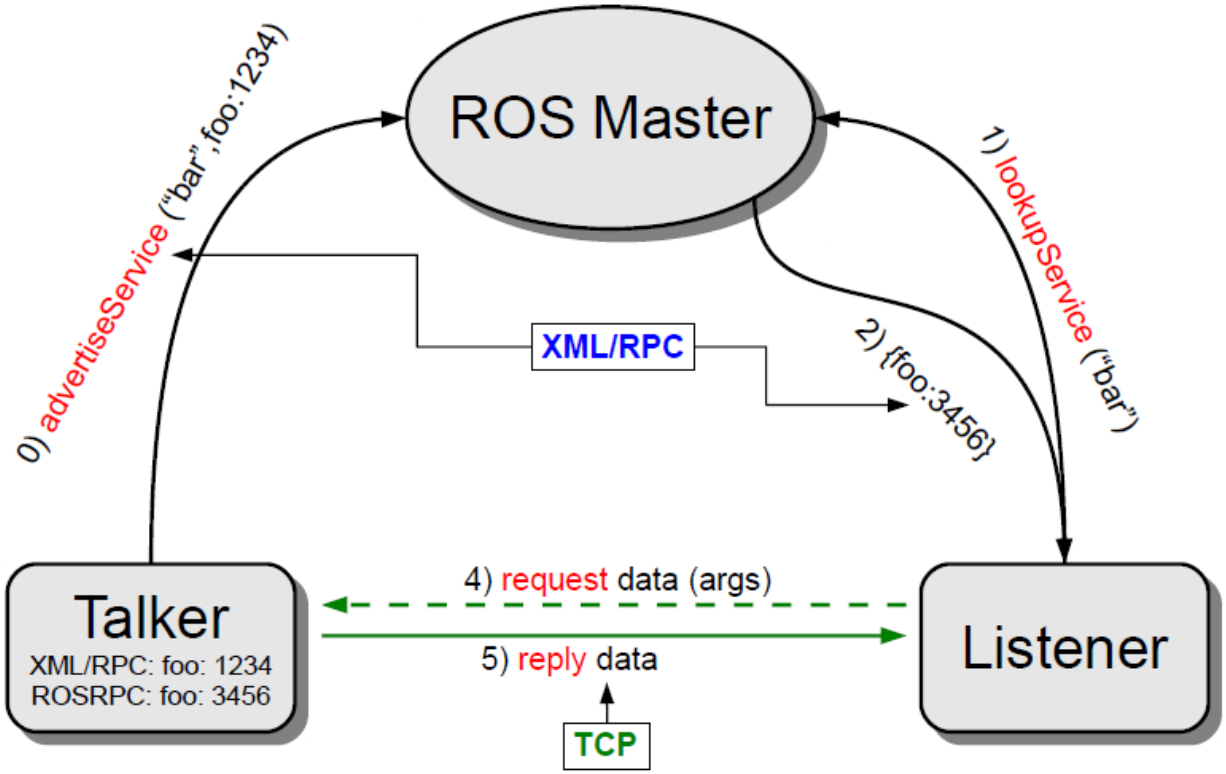
További információ:
http://wiki.ros.org/Services

RPC megvalósítása RTM környezetben
Az RTM-ben a szolgáltatások megvalósítása a CORBA RPC mechanizmusán keresztül történik. Ez egy szabványos RPC technológia, amellyel az egyes RT-Komponensek szolgáltatásokat ajánlhatnak (provide), amelyeket más komponensek igénybe vehetnek (consume). A szolgáltatások leírása az IDL (Interface Definition Language) nyelv segítségével történik. Objektum prototípusokból és szabad metódusokból építhetünk interfészeket, amelyeket a szolgáltató és a fogyasztó oldalnak egyaránt ismernie kell.

További információ:
http://www.openrtm.org/openrtm/en/content/service-port-basic
https://en.wikipedia.org/wiki/Common_Object_Request_Broker_Architecture
http://www.omg.org/gettingstarted/omg_idl.htm

Komponens keretrendszerek – Kommunikácó IV.

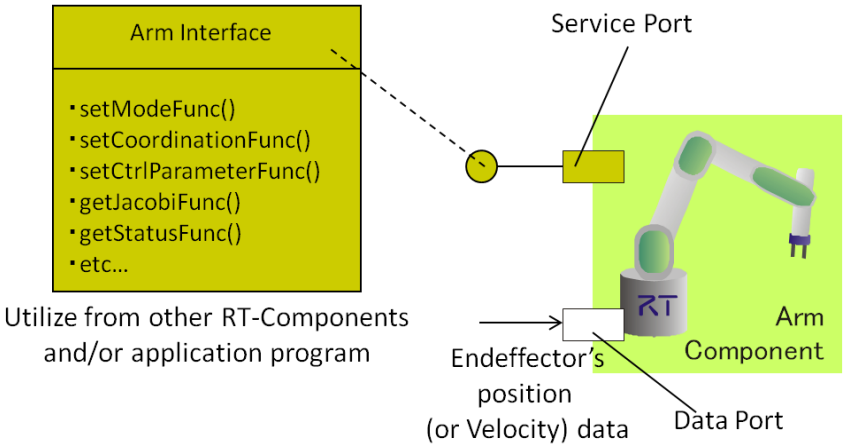
Service architektúra működése ROS környezetben



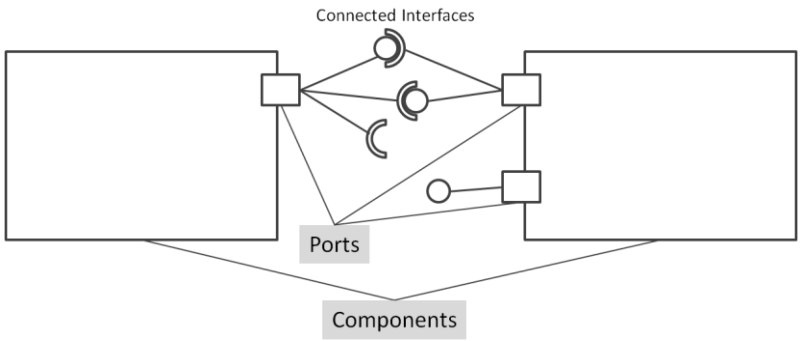
Forrás: http://barraq.github.io/fOSSa2012/slides.html#slide-17

Komponens keretrendszerek – Kommunikácó V.

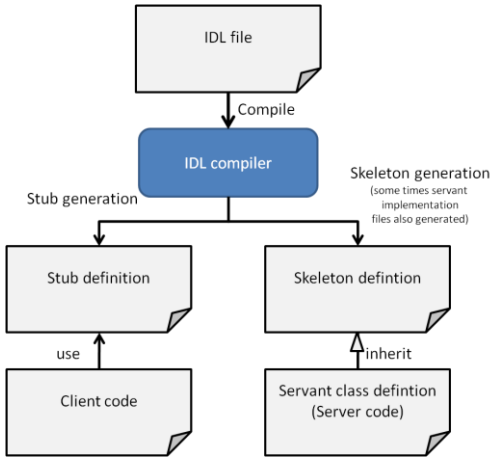
Service architektúra működése RTM környezetben



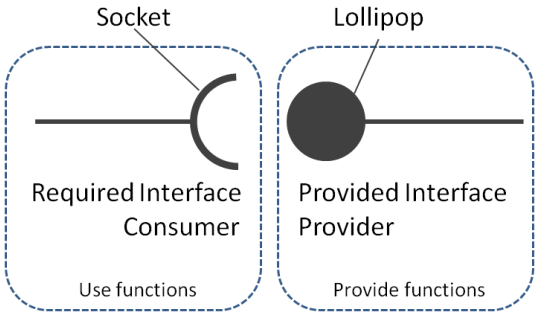
Példa a service portok tipikus alkalmazására.



Service port-on keresztül csatlakozó interfészek.



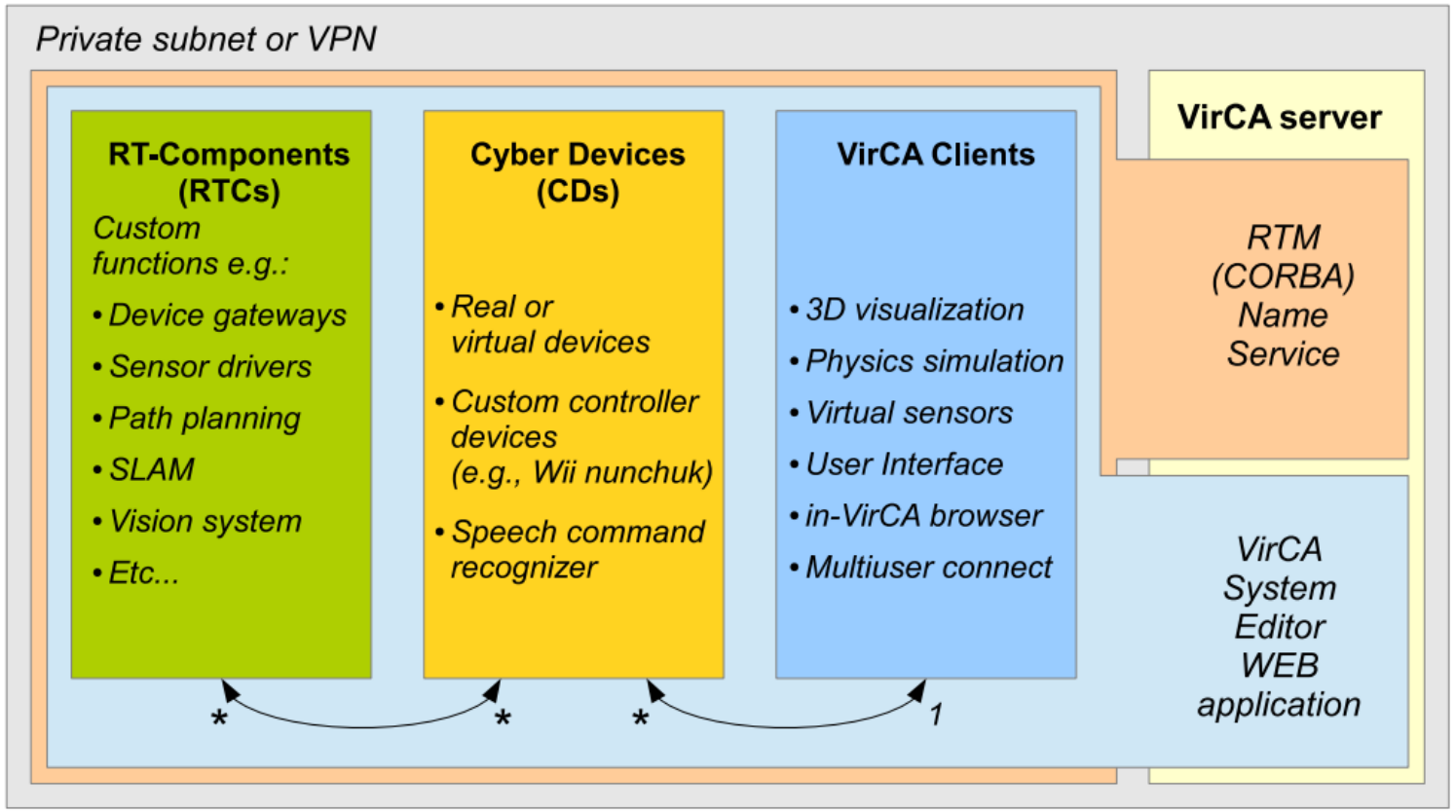
Az IDL-től az célnyelvi implementációig.



Service-ek kapcsolatának grafikus ábrázolása

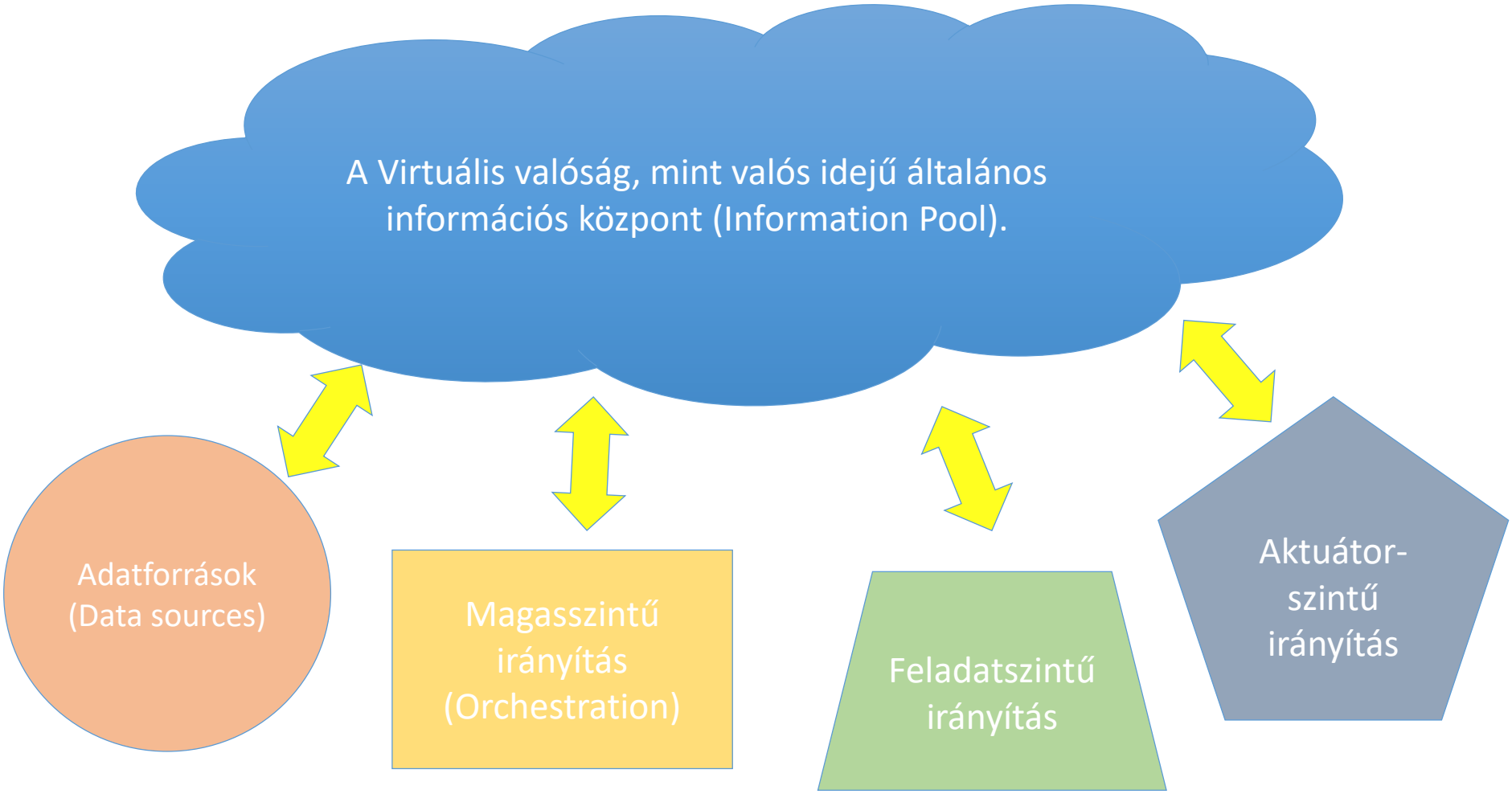
Forrás: http://www.openrtm.org/openrtm/en/content/service-port-basic

VR alapú elosztott információs közeg I.



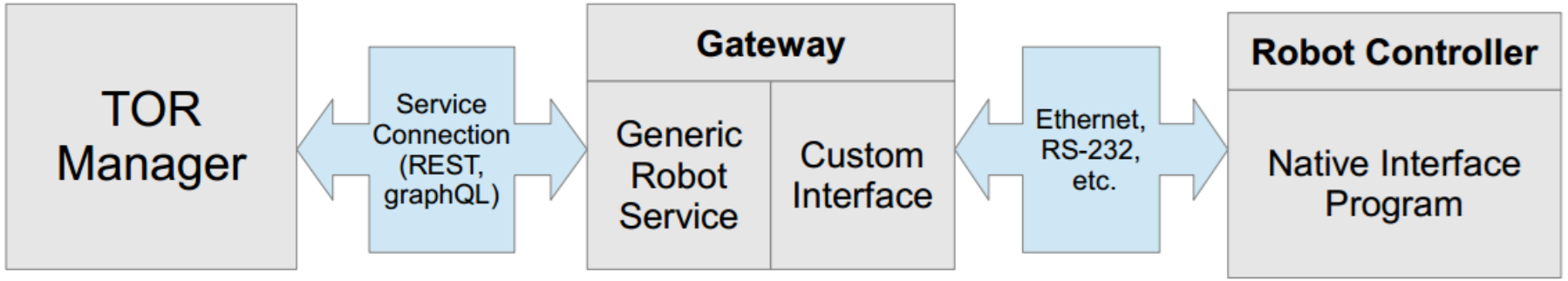
VirCA alapú elosztott szoftver-rendszer általános architektúrája.

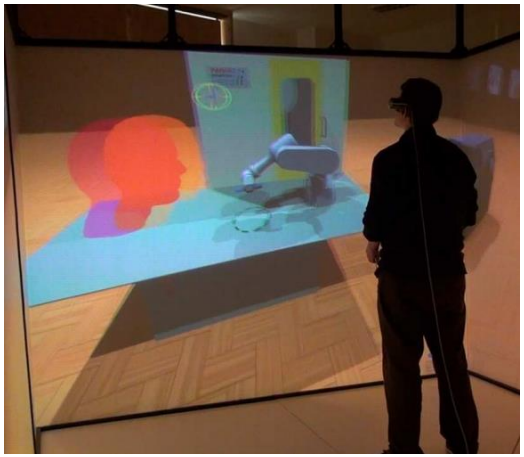
VR alapú elosztott információs közeg II.



Különböző funkciók logikai és térbeli szétcstolása

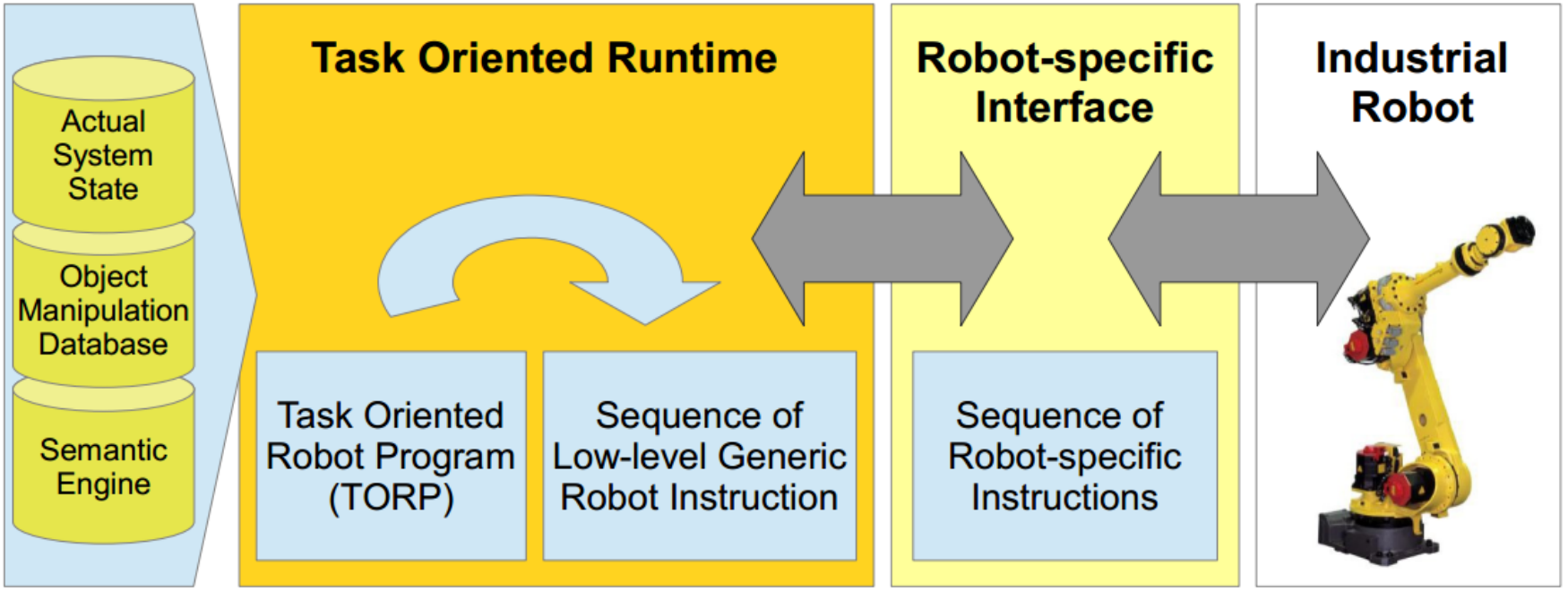
VR alapú elosztott információs közeg III.





Valós eszközök csatlakoztatása a VR alapú információs térhez.

VR alapú elosztott információs közeg IV.



Generikusan (gyártófüggetlenül) programozható robotrendszer struktúra-diagramja.

Linkek – Moduláris keretrendszerek

http://openrtm.org/
http://www.ros.org/

http://www.corba.org/
http://portals.omg.org/dds/
http://www.drdobbs.com/web-development/restful-web-services-a-tutorial/240169069

Linkek – VirCA

VirCA oldalak
http://virca.hu/
http://virca.hu/vircapedia/

Tartalomfejlesztés
http://www.sketchup.com/
http://virca.hu/vircapedia/current/index.php/SketchUp_to_VirCA_Exporter
http://virca.hu/vircapedia/current/index.php/Inserting_a_SketchUp_model_into_VirCA

Válogatott szakirodalom

[1] Y. Chen and X. Bai, “On Robotics Applications in Service-Oriented Architecture,” in *2008 The 28th International Conference on Distributed Computing Systems Workshops*, Beijing, China, 2008, pp. 551–556.

[2] N. Ando, T. Suehiro, K. Kitagaki, T. Kotoku, and Woo-Keun Yoon, “RT-middleware: distributed component middleware for RT (robot technology),” in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Edmonton, Alta., Canada, 2005, pp. 3933–3938.

[3] P. Galambos, Á. Csapó, P. Zentay, I. M. Fülöp, T. Haidegger, P. Baranyi, and I. J. Rudas, “Design, programming and orchestration of heterogeneous manufacturing systems through VR-powered remote collaboration,” *Robotics and Computer-Integrated Manufacturing*, vol. 33, pp. 68–77, Jun. 2015.

[4] P. Galambos, C. Weidig, Péter Zentay, Á. Csapó, P. Baranyi, J. C. Aurich, B. Hammann, and O. Kreylos, “Future Internet-based Collaboration in Factory Planning,” *Acta Polytechnica Hungarica*, vol. 11, no. 7, 2014.

[5] P. Galambos, P. Baranyi, and I. J. Rudas, “Merged physical and virtual reality in collaborative virtual workspaces: The VirCA approach,” in *IECON 2014 - 40th Annual Conference of the IEEE Industrial Electronics Society*, 2014, pp. 2585–2590.

[6] P. Galambos, C. Weidig, Péter Zentay, Á. Csapó, P. Baranyi, J. C. Aurich, B. Hammann, and O. Kreylos, “VirCA NET: A Collaborative Use Case Scenario on Factory Layout Planning,” presented at the 3rd IEEE International Conference on Cognitive Infocommunications, Kosice, Slovakia, 2012, pp. 467–468.

[7] P. Galambos and P. Baranyi, “VirCA as Virtual Intelligent Space for RT-Middleware,” in *2011 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, Budapest, Hungary, 2011, pp. 140–145.