Metadata of the chapter that will be visualized in SpringerLink

Book Title	Recent Advances in Robotics and Automation	
Series Title	7092	
Chapter Title	Application of Knowledge Driven Mobile Robots for Disassembly Tasks	
Copyright Year	2013	
Copyright HolderName	Springer-Verlag Berlin Heidelberg	
Corresponding Author	Family Name	**Koppensteiner**
	Particle	
	Given Name	**Gottfried**
	Suffix	
	Division	Institute of Automation and Control
	Organization	Vienna University of Technology
	Address	Vienna, Austria
	Email	koppensteiner@acin.tuwien.ac.at
Author	Family Name	**Krofitsch**
	Particle	
	Given Name	**Christoph**
	Suffix	
	Division	
	Organization	Practical Robotics Institute Austria
	Address	Vienna, Austria
	Email	krofitsch@pria.at
Author	Family Name	**Hametner**
	Particle	
	Given Name	**Reinhard**
	Suffix	
	Division	Institute of Automation and Control
	Organization	Vienna University of Technology
	Address	Vienna, Austria
	Email	hametner@acin.tuwien.ac.at
Author	Family Name	**Miller**
	Particle	
	Given Name	**David P.**
	Suffix	
	Division	Schools of AME and Computer Science
	Organization	University of Oklahoma
	Address	Norman, OK, USA
	Email	dpmiller@ou.edu
Author	Family Name	**Merdan**
	Particle	
	Given Name	**Munir**

Suffix	
Division	Institute of Automation and Control
Organization	Vienna University of Technology
Address	Vienna, Austria
Email	merdan@acin.tuwien.ac.at

Abstract

Considering the disassembly as a vital and prospective industry domain, we use the mobile robots to automate the disassembly process. In our system, each mobile robot has particular skills and is supervised by an agent with related objectives and knowledge. An agent has an ontology-based world model, which is responsible to maintain the knowledge about the robot's activities in relation to its environment as well as to its underlying software parts. The ontology is used to represent a specification of an agent's domain knowledge. The system functionality is tested with three mobile robots having a task to disassemble a particular Lego construct. Different rule-engines were benchmarked in order to enhance the systems performance.

Layout: T1 Standard SC | Book ID: 272920_1_En | Book ISBN: 978-3-642-37386-2
Chapter No.: 24 | Date: 30-3-2013 | Page: 1/11

Application of Knowledge Driven Mobile Robots for Disassembly Tasks

Gottfried Koppensteiner, Christoph Krofitsch, Reinhard Hametner, David P. Miller and Munir Merdan

AQ1

Abstract Considering the disassembly as a vital and prospective industry domain, we use the mobile robots to automate the disassembly process. In our system, each mobile robot has particular skills and is supervised by an agent with related objectives and knowledge. An agent has an ontology-based world model, which is responsible to maintain the knowledge about the robot's activities in relation to its environment as well as to its underlying software parts. The ontology is used to represent a specification of an agent's domain knowledge. The system functionality is tested with three mobile robots having a task to disassemble a particular Lego construct. Different rule-engines were benchmarked in order to enhance the systems performance.

Based on "Knowledge Driven Mobile Robots Applied in the Disassembly Domain", by Gottfried Koppensteiner, Reinhard Hametner, Rene Paris, Alejandro Moser Passani, and Munir Merdan which appeared in the Proceedings of the 5th International Conference on Automation, Robotics and Applications (ICARA 2011). © 2011 IEEE.

G. Koppensteiner (✉) · R. Hametner · M. Merdan
Institute of Automation and Control, Vienna University of Technology, Vienna, Austria
e-mail: koppensteiner@acin.tuwien.ac.at

R. Hametner
e-mail: hametner@acin.tuwien.ac.at

M. Merdan
e-mail: merdan@acin.tuwien.ac.at

C. Krofitsch
Practical Robotics Institute Austria, Vienna, Austria
e-mail: krofitsch@pria.at

D. P. Miller
Schools of AME and Computer Science, University of Oklahoma, Norman, OK, USA
e-mail: dpmiller@ou.edu

G. Sen Gupta et al. (eds.), *Recent Advances in Robotics and Automation*, Studies in Computational Intelligence 480, DOI: 10.1007/978-3-642-37387-9_24,

Layout: T1 Standard SC | Book ID: 272920_1_En | Book ISBN: 978-3-642-37386-2
Chapter No.: 24 | Date: 30-3-2013 | Page: 2/11

1 Introduction

Disassembly has become a vital industry process due to the increasing necessity of optimizing resource usage. Currently, disassembly processes are partially automated only in a small set of cases such as: single use cameras [1], PCs [2], printed circuit boards as well as LCD monitors [3]. The main limiting factors are: non-uniformity of returned product models creating great uncertainty in the system control and structural configuration, etc. [4]. The rigid character and weak adaption capabilities of current implementations, due in part to their use of centralized hierarchical control structures, limits their ability to respond efficiently and effectively to dynamic changes.

Mobile robots offer new possibilities for flexible disassembly [5]. The state-of-the-art of mobile robot technology and predictions of future development give a clear view that mobile robots are going to be an essential part of every manufacturing process in future [6]. Mobile robots can support the system's flexibility and capability to handle dynamic changes. However, one of the obstacles to wider adoption of the mobile robots in the industry is the engineering cost to integrate mobile robots into systems. The field of robotics should develop in such a way to reduce the programming requirements and to increase the flexibility of mobile robots for different tasks [7]. In addition, mobile robots for disassembly should be (a) intelligent in the sense of path planning and able to communicate with other robots, (b) cooperative with other (stationary or mobile) robots, and (c) able to form a disassembly multi agent system, which is one of the future possibilities for reducing disassembly costs. Moreover, in order to avoid difficulties in communication in a heterogeneous robot environment, where each robot has its own kinematic structure and programming language, etc., it is necessary to develop standardized communication protocols and methods [5].

To cope with these requirements, we propose a knowledge-intensive multi-agent robot system, which enables ontology-based communication and cooperation among a set of autonomous and heterogeneous units (agents). In our system each agent supervises one particular mobile robot and, related to the robot's skills, has its own objectives and knowledge. In this context, ontologies allow the explicit specification of an agent's domain of application, increasing the level of specification of knowledge by incorporating semantics into the data and promoting knowledge exchange between agents in an explicitly understandable form [8]. An ontology based product model is used [9] to link product designs, disassembly planning and scheduling processes, as well as required disassembly equipment, possessed by a particular mobile robot, in a way that enables automatic reasoning as well as wide data integration. Consequently, on the one side, a vision system can use this model to reason about the content of a captured image. On the other side, an agent controlling a mobile robot can extract required disassembly information from this model to select and perform the necessary actions [10]. The architecture is based on agents that have a rule-based behavior. Rules are

Layout: T1 Standard SC | Book ID: 272920_1_En | Book ISBN: 978-3-642-37386-2
Chapter No.: 24 | Date: 30-3-2013 | Page: 3/11

considered as if–then statements applied to the knowledge base. The application of this kind of decision-making mechanism supports a knowledge capture in a more modular and explicit way.

2 Agent Architecture

In previous work [11–13] a generic agent architecture was developed to facilitate the design of multi-agent control systems. This architecture clearly separates the control software into two layers: the high level control (HLC) and the low level control (LLC). The LLC layer is in charge of controlling the hardware directly. It is responsible for performing all necessary operations in real-time and is based on the IEC 61499 Standard [14]. The HLC layer is responsible for more complex tasks such as coordination, monitoring or diagnostic, which might require a longer computation time. This allows using one HLC-Architecture with different LLCs caused by heterogeneous mobile robots. In the rest of the chapter, we will present the structure and functionality of the HLC adapted for the control of and negotiation between mobile robots. To enable agents to understand the transmitted messages between the LLC and HLC, a representation data type is as well as the *MessageContent* concept and its sub-concepts is included in the agent's ontology [15]. Figure 1 shows the architecture of the system which provides the HLC for the mobile robots, with focus on the interplay of the used technologies.

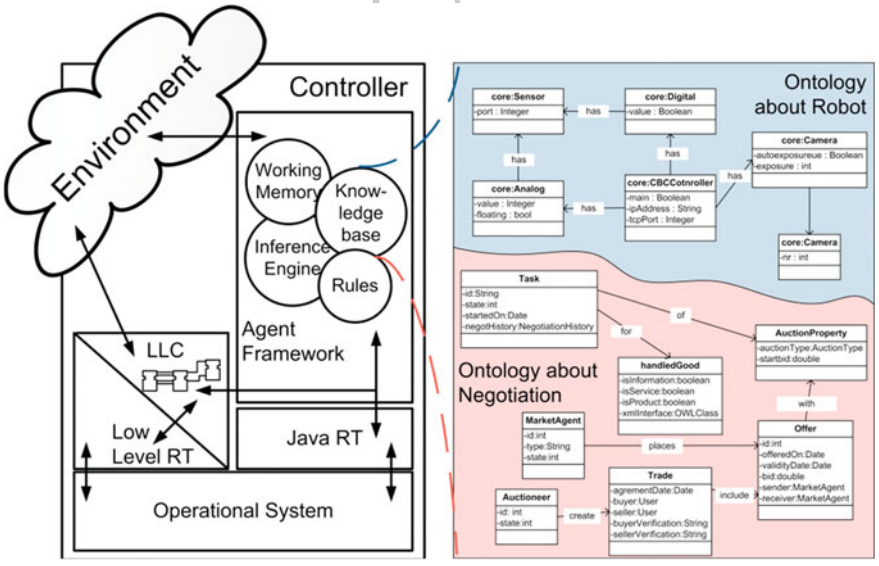

Fig. 1 Agent architecture includes the HLC and LLC overview [23] applied on mobile robots in the disassembly (negotiation) domain

The whole system environment is based on an ontology which is copied to each of the controllers. The framework uses its ontology for multiple purposes. First of all, it serves as the basis for the rule based control system, but additionally, it serves as a place for configuring the robot and the intercommunication. The ontology describes typical components from robots (e.g. sensors, motors) as well as the robots communication including coordination, cooperation and negotiation) with all necessary protocols. The ontology can be easily extended to support application specific needs, e.g. for mapping disassembly structures. This supports the agents with the necessary domain knowledge about the robots and the negotiation mechanisms between them in order to provide the robots with a framework for collaboration. The knowledge of the HLC is defined in rules which are different for each robot type. The knowledge base is periodically updated. This is realized through an updating process which is performed between the specific system-libraries of the controller and the ontology.

3 System Integration

The High Level Robot Control is implemented on the CBCv2 [16] which includes an ARM 7 based DAQ/Motor control system, an ARM 9-based CPU/Vision processor running Linux, an integrated color display as well as a touch screen. It is being used by thousands of middle and high school students in the educational robotics program Botball [17]. Due to the aim of the sparkling science initiative [18], it was one major reason to take a controller system which is well-suited for a collegiate robotics lab. Therefore, the Atmel ARM7 32 bit MCU running at 48 MHz, because of its speed, availability, and wide range of embedded communication and I/O ports, including 16 10-bit ADCs, is good enough to meet the requirements for a robotics Lab. On the other side, the embedded Linux and reloadable firmware provide a complete package for the enhancement with agent and rule based systems. The CBCv2 is a USB host (allowing the use of standard cameras, mass storage and network interfaces) and can also be used as a USB device for software downloads. At the USB port a Wi-Fi Stick can be used; this provides a good possibility for the communication between mobile robots.

3.1 Framework

On top of the Linux-based robot controller, explained in the previous section, the CBCJVM [19] is used as the system-library. CBCJVM uses a lightweight Java Virtual Machine (JVM), in particular the JamVM, which is designed for embedded platforms which makes it perfect for the use on the CBC. It includes libraries for working with the low level components of the CBC, including reading sensors and controlling motors, which are used by the updating process to synchronize the

Layout: T1 Standard SC | Book ID: 272920_1_En | Book ISBN: 978-3-642-37386-2
Chapter No.: 24 | Date: 30-3-2013 | Page: 5/11

ontology objects representing low-level components. In this process, the controller's own components mapped in the ontology are synchronized with the values from the system-libraries (e.g. sensor values) at fixed update intervals. On top of this, the JADE-Leap platform takes place. JADE [20] simplifies the implementation of multi-agent systems through a middle-ware that complies with the FIPA specifications [21]. The LEAP libraries allow obtaining a FIPA-compliant agent platform with reduced footprint and compatibility with mobile Java environments. Altogether, this is the host for the Agents, which finally control the robots. On the one side, they are responsible for listening for ontology changes and activating the rule engine appropriately, on the other side they handle the intercommunication between other robots, which is further described in the next section. This architecture enables writing rule-based behaviors using the full knowledge from the ontology, one of the key aspects of this framework. The rule-based behaviors are written in different rule-based languages and are fit to the mechanical possibilities of the robot and the tasks the robot is intended for. Currently, the applicable rule engines are Jess and OPSJ.

3.2 Agent Communication

The agent communication ensures the possibility to share ontology values between different controllers. That enables defining rules for foreign ontology values or using knowledge from other robots in order to provide reliable data for robot collaboration or about a deal within the negotiation process.

Figure 2 shows two different negotiation procedures in the disassembly process. The task allocation with auctions starts with a "need for some service". This order could be transmitted to only one robot which is able to fulfill the requested operation, or as broadcast within an auction procedure to evaluate the best situated robot. In order to ensure that every agent in the system talks about the same part of interest, parts of the ontologies could be transmitted within these messages (see message "Object: yellow part") and in order to be able to reason about it, mapped to the agent's ontology. If more than one service is necessary for this operation, the requesting agent starts one "auction" (request) for each service. Every robot which is able to perform some particular service sends a bid or the offered service back. After all subjects of the deal are negotiated, a deal-commitment is sent and the operation is started. Within such operations, spread between different robots, coordination might be necessary in order to schedule each step of the operation. This is done by negotiation about the necessary operations. In this procedure two or more robots, working together to solve one requested task, exchange messages about necessary steps. Therefore a fast reaction time is necessary: in the case of the LLC to be able to react quickly to sensor values (for instance to avoid crashes); in the HLC to be able to reason as fast as possible about incoming messages.

Layout: T1 Standard SC | Book ID: 272920_1_En | Book ISBN: 978-3-642-37386-2
Chapter No.: 24 | Date: 30-3-2013 | Page: 6/11

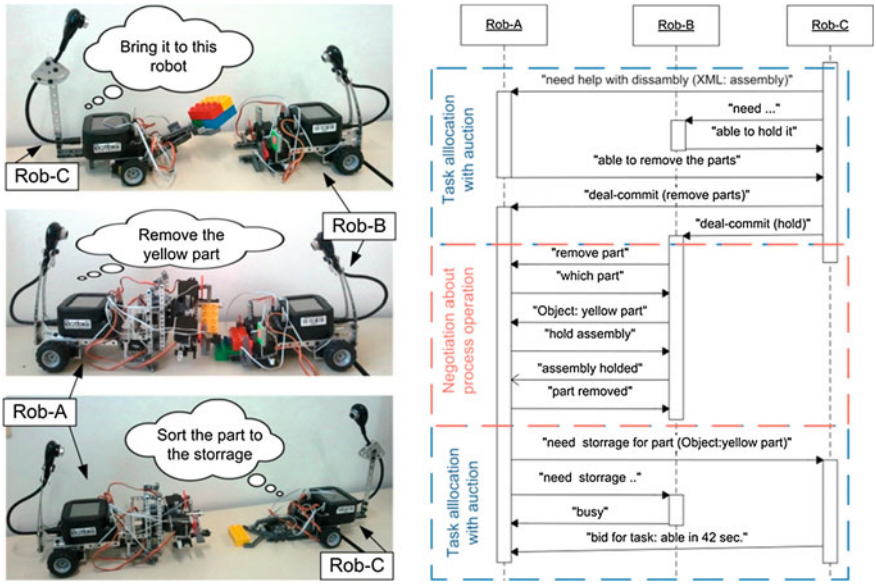

Fig. 2 Robot negotiation

4 System Implementation

Due to memory constraints of the CBCv2, it was not possible to use the Protégé ontology API [22] at runtime. Instead, the ontology is translated into Java classes and objects which get stored to a file at build time. The access to the serialized file is then possible on the CBCv2. The mapping is done in two stages: firstly, the classes are designed in the ontology and are converted into Java classes. Secondly, after the generated classes are compiled, the objects contained in the ontology are read, instantiated, and written to a file via Java serialization. At runtime, the objects are read into the framework via the ontology interface, and added to the rule engine's fact base.

4.1 Customizability and Flexibility

The HLC framework implementation was focused on two important characteristics: (1) great amount of customizability for being applicable in various fields and tasks, and (2) flexibility for different software- and hardware-environments through exchangeability. Beside definition of application rules, the customizability was realized by (1) providing almost unlimited ontology extension with new classes and instances, (2) implementing and executing new software agents, and (3) adding user-commands for executing specific pieces of code from rules or

Layout: T1 Standard SC | Book ID: 272920_1_En | Book ISBN: 978-3-642-37386-2
Chapter No.: 24 | Date: 30-3-2013 | Page: 7/11

command line. This is possible through holding the framework configuration and interfaces in one single object, which is globally accessible. In terms of flexibility, the following framework components can be fully exchanged:

- Multi-Agent System: Instead of JADE, another FIPA-compliant Agent Platform might be used.
- Rule Engine: The whole rule-based system including inference engine and rule language can be exchanged. Currently, Jess and OPSJ might be used.
- Hardware interface: Instead of using the CBCv2, another robot controller might be used by implementing the low-level library connection.

In this system, plugins representing single components can be implemented in Java projects containing classes implementing specific Java interfaces defined in the core project of the framework. The usage of plugins can be statically configured in a specially implemented configuration format (e.g. when changing the right plugin class in the configuration file, OPSJ rule engine can be used instead of Jess without any other efforts). Example rules for both rule engines can be found in the next chapter in the context of performance measurements.

5 Performance Benchmarking

When implementing this framework in its first version, Jess was the only applicable rule engine. In most cases, the rule firing time was insufficient due to the limited resources of the CBC, and some measurements showed that the engine used in Jess consumes most of the controller's resources. Considering the importance of fast reaction on particular events it was necessary to investigate possible system improvements. Trying out another rule engine might solve this problem and this was actually the reason to implement the plugin into the system which was introduced the section before. For comparing Jess and OPSJ rule engines, two different performance benchmarks are used and their results presented later in this section. Both tests are executed in the framework environment, which means that JADE and the ontology might have an effect on the results.

5.1 Methodology

The first benchmark is separated into two parts. It is intended for measuring the time the different rule engines consume to load Java objects into their working memory. After loading, the allocated heap memory is measured. Therefore, this benchmark shows the data loading time and the heap memory consumption dependent on the number of objects to load. Every object's state consists of ten integer variables and one string variable; the maximum heap memory on the CBC is limited to 32 megabytes. The execution is planned as follows: Each run loads a

Layout: T1 Standard SC | Book ID: 272920_1_En | Book ISBN: 978-3-642-37386-2
Chapter No.: 24 | Date: 30-3-2013 | Page: 8/11

number of objects, beginning with 10 for the first run up to 5,000 for the last run. After loading has finished in each run, the allocated heap memory will be measured. Every run will be iterated ten times and averaged for the result.

The second benchmark illustrates the optimal way of using a rule engine in the framework. It simulates the usage of one analog sensor in the framework, which means that the sensor will be read periodically. When the sensor's value is in a specific range, the time it takes to fire a rule, whose left-hand side matches this range, will be measured (reaction time). In particular, the reaction time consists of two parts: (1) the time for updating the appropriate rule engine fact with the new sensor value and (2) the time the rule engine takes to fire the rule. Because the sensor's value changes continually, every update leads to an invocation of the rule engine. This could overload the CBC and causes longer rule engine reaction times. This benchmark measures the overall reaction time for different update intervals and reveals the most efficient update interval configurations for the framework. The execution is planned as follows: Each run, using a specific update period beginning at 120 ms for the first run down to 0 ms for the last run, measures the reaction time 250 times, i.e. fires the rule 250 times. Figure 3 shows the rules for this performance benchmark in Jess- and OPSJ language. In the right-hand side, a static method will be called, which measures the time and prepares for the next iteration.

5.2 Results

The results of the first benchmark are presented as follows (Fig. 4):

The object loading time is roughly linear for both rule engines; for better visualization the loading time is scaled logarithmically. OPSJ is significantly faster than Jess, on average by a factor of 38. The heap memory plot is smoothed slightly in order to compensate side effects caused by JADE or other disturbance. Here is OPSJ performing better as well, the average memory savings compared to Jess are 2.6 Megabytes.

```
(defrule testrule
  (AnalogSensor {value > 500})
  =>
  (call main.Main ruleFired)
)
```

```
rule testrule
if {
    d: AnalogSensor (
      d.getValue() > 500
    );
} do {
    Main.ruleFired();
}
```

Fig. 3 Rule used in reaction time performance benchmark: Jess (*left*) and OPSJ (*right*)

Layout: **T1 Standard SC** Book ID: **272920_1_En** Book ISBN: **978-3-642-37386-2**
Chapter No.: **24** Date: **30-3-2013** Page: **9/11**

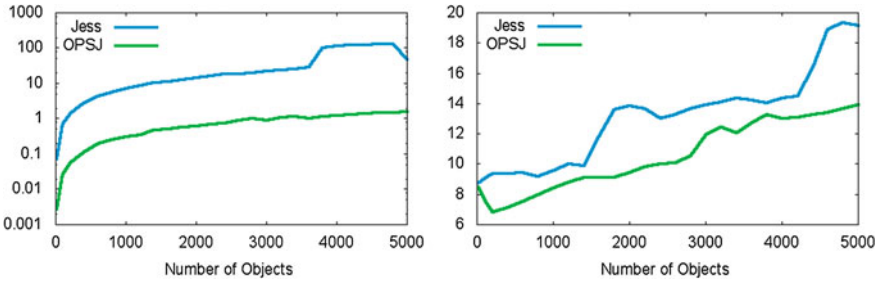

Fig. 4 Results of the data performance benchmark: object loading time on number of objects (*left*) and allocated heap memory on number of objects (*right*)

Figure 5 shows the results of second performance benchmark. The 250 measurements, which were taken in each run, are visualized as a boxplot. They illustrate the range of reaction times which are possible for a specific update interval. As can be seen, every range goes from nearly zero (min) up to at least the update interval (max). This is explained by the randomness of the sensor's value: When the value passes the threshold, the next update could be immediate or long time off.

As expected, a low update interval (10 and 0 ms) when using Jess takes the controller to overload. In the graph, this can be seen in the raising maximum at 10 ms which indicates unstable reaction time. At 0 ms, even the average reaction time raises. In case of OPSJ, reaction time does not increase significantly with low update interval. Note, that this benchmark simulates just one analog sensor which is insufficient in most robotic applications. When using an appropriate number of sensors and actuators, both rule engines might cause serious performance issues with low update intervals; Jess more than OPSJ.

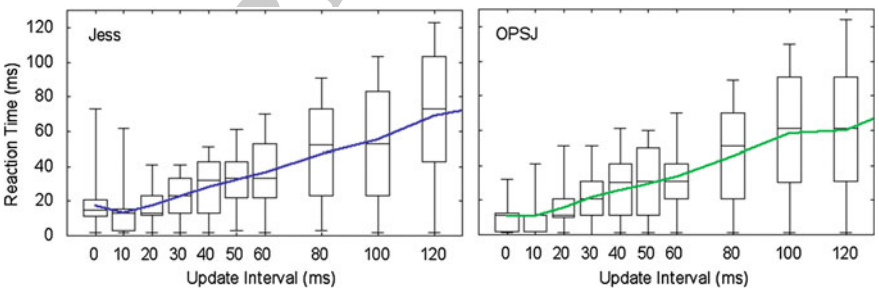

Fig. 5 Results of the reaction time performance benchmark: reaction time on update interval; visualization of raw data as boxplots and averaged as line

6 Conclusion

In this work, an intelligent MAS based on ontologies for a distributed mobile robot system applied in the disassembly domain was presented. The architecture provides robots, which are intelligent in the sense of path and task planning as well as cooperative with other (stationary or mobile) robots. They are also able to communicate with other robots and to coordinate disassembly tasks using negotiations. In order to face the insufficient reaction time due to the limited resources of the CBC a customizable architecture was developed, which allows the replacement of agent platforms, rule engines or hardware devices. Currently, Jess and OPSJ are the applicable rule engines, which differ significantly in their resource requirements. Jess takes a lot more time for data loading and allocates more heap memory than OPSJ. Furthermore, the lightweight OPSJ engine fires rules faster than the Jess engine, which mostly comes out when updating sensor and actuator values in low interval. This clearly points out the benefits of OPSJ in allocated memory and processing time, which both are the main limiting factors on mobile controllers. In this work the HLC was benchmarked. In order to provide the agents with faster reactive behaviors for sensors and motors a similar set of tests and evaluations of the LLC should be performed.

Acknowledgments The authors would like to acknowledge the support by the Sparkling Science program, an initiative of the Austrian Federal Ministry of Science and Research. We also want to thank all partners involved in the DISBOTICS Project, especially the students at Vienna Institute of Technology (TGM), department for Information-Technology, and the KISS Institute of Practical Robotics.

References

1. M. Matsumoto, Business frameworks for sustainable society: a case study on reuse industries in Japan. J. Cleaner Prod. **17**(17), 1547–1555 (2009)
2. F. Torres et al., Automatic PC disassembly for component recovery. Int. J. Adv. Manufact. Technol. **23**(1), 39–46 (2004)
3. H. Kim, S. Kernbaum, G. Seliger, Emulation-based control of a disassembly system for LCD monitors. Int. J. Adv. Manuf. Technol. **40**(3), 383–392 (2009)
4. J.R. Duflou, G. Seliger, S. Kara, Y. Umeda, A. Ometto, B. Willems, Efficiency and feasibility of product disassembly: a case-based study. CIRP, Manufact. Technol. **57**(2), 583–600 (2008)
5. P. Kopacek, B. Kopacek, Intelligent, flexible disassembly. Int. J. Adv. Manufact. Technol. **30**(5), 554–560 (2006)
6. F.M. Asl, A.G. Ulsoy, Y. Koren, *Dynamic Modeling and Stability of the Reconfiguration of Manufacturing Systems*. Technical report (University of Michigan, 2001)
7. A. Lazinica, B. Katalinic, Self-organizing multi-robot assembly system. Int. Symp. Robot. **36**, 42 (2005)
8. M. Merdan, *Knowledge-based Multi-Agent Architecture Applied in the Assembly Domain.* PhD Thesis, Vienna University of Technology, (2009), http://www.ub.tuwien.ac.at/diss/AC05040230.pdf

Layout: T1 Standard SC | Book ID: 272920_1_En | Book ISBN: 978-3-642-37386-2
Chapter No.: 24 | Date: 30-3-2013 | Page: 11/11

9. K. Kyoung-Yun, D.G. Manley, H. Yang, Ontology-based assembly design and information sharing for collaborative product development. Comput. Aided Des. **38**(12), 1233–1250 (2006)
10. M. Merdan, W. Lepuschitz, T. Meurer, M. Vincze, in *Towards Ontology-Based Automated Disassembly Systems*. IEEE international conference on industrial electronics control and instrumentation, (2010), pp. 1386–1391.G
11. G. Koppensteiner, M. Merdan, W. Lepuschitz, I. Hegny, in *Hybrid Based Approach for Fault Tolerance in a Multi-Agent System*. IEEE/ASME international conference on advanced intelligent mechatronics, (Singapore, 2009)
12. M. Merdan, M. Vallee, W. Lepuschitz, A. Zoitl, Monitoring and diagnostics of industrial systems using automation agents. Int. J. Prod. Res. **49**(5), 1497 (2011)
13. M. Vallee, H. Kaindl, M. Merdan, W. Lepuschitz, E. Arnautovic, P. Vrba, in *An Automation Agent Architecture With a Reflective World Model in Manufacturing Systems*. IEEE international conference on systems, man, and cybernetics (SMC09), (San Antonio, Texas, USA, 2009)
14. A. Zoitl, *Real-Time Execution for IEC 61499*. ISA-o3neidaA, USA, ISBN: 978193439-4274 (2009)
15. M. Merdan, W. Lepuschitz, I. Hegny, G. Koppensteiner, in *Application of a Communication Interface Between Agents and the Low Level Control*. Proceedings of the 4th international conference on autonomous robots and agents, Wellington, New Zealand, 2009
16. D.P. Miller, M. Oelke, M.J. Roman, J. Villatoro, C.N. Winton, in *The CBC: A LINUX-based low-cost mobile robot controller*. IEEE international conference on robotics and automation (ICRA), pp. 4633–4638, 3–7 May 2010
17. KISS Institute of Practical Robotics, *The Botball Season*. http://www.botball.org, Accessed May 2011
18. Sparkling Science, BMWF. http://www.sparklingscience.at/en. last viewed June 2011
19. B. McDorman, B. Woodruff, A. Joshi, J. Frias, in *CBCJVM: Applications of the Java Virtual Machine with Robotics*. Global conference on educational robotics, Edwardsville (2010)
20. Telecom Italia Labs, *Java Agent Development Framework*. http://jade.tilab.com/. Accessed March 2011
21. The Foundation for Intelligent Physical Agents, *F*IPA Specifications. http://www.fipa.org/specifications/index.html. last viewed July 2011
22. Stanford Medical Informatics, *Protégé Ontology Editor*. Stanford University. Protégé Website. http://protege.stanford.edu. Accessed May 2011
23. G. Koppensteiner, M. Merdan, I. Hegny, W. Lepuschitz, S. Auer, B. Groessing, in *Deployment of an Ontology-Based Agent Architecture on a Controller*. 8th IEEE international conference on industrial informatics, Japan, 2010
24. Sandia National Laboratories, *Jess: The Rule Engine for the JavaTM Platform*. Available at: http://herzberg.ca.sandia.gov/. last visited May 2011

AQ2

Author Query Form

Book ID :	**272920_1_En**
Chapter No.:	**24**

Springer
the language of science

Please ensure you fill out your response to the queries raised below and return this form along with your corrections

Dear Author
During the process of typesetting your chapter, the following queries have arisen. Please check your typeset proof carefully against the queries listed below and mark the necessary changes either directly on the proof/online grid or in the 'Author's response' area provided below

Query Refs.	Details Required	Author's Response
AQ1	Please confirm the corresponding author is correctly identified and amend if necessary.	
AQ2	Reference [24] is given in list but not cited in text. Please cite in text or delete from list.	

MARKED PROOF

Please correct and return this set

Please use the proof correction marks shown below for all alterations and corrections. If you wish to return your proof by fax you should ensure that all amendments are written clearly in dark ink and are made well within the page margins.

Instruction to printer	*Textual mark*	*Marginal mark*
Leave unchanged	··· under matter to remain	
Insert in text the matter indicated in the margin		New matter followed by or
Delete	through single character, rule or underline or through all characters to be deleted	or
Substitute character or substitute part of one or more word(s)	through letter or through characters	new character or new characters
Change to italics	under matter to be changed	
Change to capitals	under matter to be changed	
Change to small capitals	under matter to be changed	
Change to bold type	under matter to be changed	
Change to bold italic	under matter to be changed	
Change to lower case	Encircle matter to be changed	
Change italic to upright type	(As above)	
Change bold to non-bold type	(As above)	
Insert 'superior' character	through character or where required	or under character e.g. or
Insert 'inferior' character	(As above)	over character e.g.
Insert full stop	(As above)	
Insert comma	(As above)	
Insert single quotation marks	(As above)	or and/or or
Insert double quotation marks	(As above)	or and/or or
Insert hyphen	(As above)	
Start new paragraph		
No new paragraph		
Transpose		
Close up	linking characters	
Insert or substitute space between characters or words	through character or where required	
Reduce space between characters or words	between characters or words affected	