10401 – Extended Abstract
The Task-State Coordination Pattern, with applications in Human-Robot-Interaction
– Dagstuhl Seminar –

Ingo Lütkebohle[1], Julia Peltason[1], Britta Wrede[1] and Sven Wachsmuth[1,2]

[1] Bielefeld University, Applied Informatics Group
Postfach 100 131, 33501 Bielefeld, Germany
{iluetkeb,jpeltaso,bwrede}@techfak.uni-bielefeld.de
[2] Bielefeld University, Central Lab Facilities
Postfach 100 131, 33501 Bielefeld, Germany
swachsmu@techfak.uni-bielefeld.de

Abstract. We consider interaction a powerful enabling technology for robots in human environments. Besides taking commands or reporting, many other uses, such as interactive learning, are already being explored. However, HRI also poses systems engineering challenges that may hinder its adoption. To address these, we advocate a general coordination pattern for task execution: The Task-State Pattern. Crucially, it separates interaction coordination from task-level control, thus enabling independent, but integrated, development.
In the pattern, tasks are represented using both a general, re-usable task coordination model and a task-type dependent specification. We have introduced a coordination model rich enough to support a powerful user experience, but still general enough to accomodate a variety of tasks, thus simplifying architecture and integration. Furthermore, because it is re-used in many places, it provides an attractive target for tool support.

Keywords. human-robot-interaction, software architecture, design pattern, dialog

1 Introduction

For robots in human environments, interaction with the humans can be both a necessity and an aid mechanism. Generally accepted as necessary are the ability to take commands from a human and to report at least the success of a task, with more detail often a help. On top of that, interaction can also be an aid, i.e. when the robot is missing information or cannot do the task alone.

From a software engineering point of view, such interaction requires that there is a detailed exchange of information between the interaction management parts and those parts that are carrying out tasks. At the very least, commands and results must be communicated back and forth. However, to keep changes to

Dagstuhl Seminar Proceedings 10401
Learning, Planning and Sharing Robot Knowledge for Human-Robot Interaction
http://drops.dagstuhl.de/opus/volltexte/2011/2931

the various parts from interfering and to generally facilitate effective development, it is desirable to keep the different parts of a system independent as much as possible. These somewhat conflicting goals can be reconciled through the use of modules with well-defined interfaces [1]. In robotics, this approach is already widely applied for middleware frameworks in robotics (cf. [2, 3]) and has been suggested as a good approach for general system construction [4, 5].

The Task-State pattern represents such an interface, to provide a general coordination mechanism for robot tasks. It is a proven engineering practice which has been extracted through repeated analysis, design and application steps. It and the proposals it is based on are in wide use in a diverse range of robotic systems [6, 7, 8, 9, 10, 11].

In the following, we will shortly introduce an example scenario encompassing HRI and manipulation and then present the pattern in this context. Afterwards, references to further case studies will be given.

2 The "Curious Robot" scenario

The "Curious Robot", shown in figure 1, is an experiment in mixed-initiative, multi-modal interaction. It integrates a bimanual manipulator and a humanoid robot torso into a single software system, with the former responsible for manipulation (e.g., picking up objects) and the latter for interaction (e.g. gaze feedback on the attended object). See `http://www.youtube.com/watch?v=D8Q8Udh7CMg` for an interaction demo.

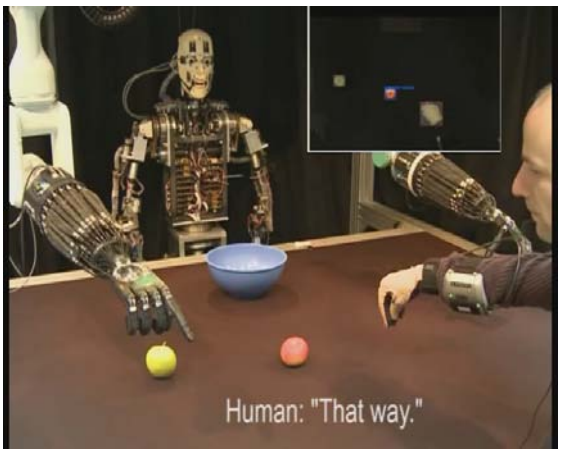


Fig. 1. Interaction example, the human demonstrates a grip.

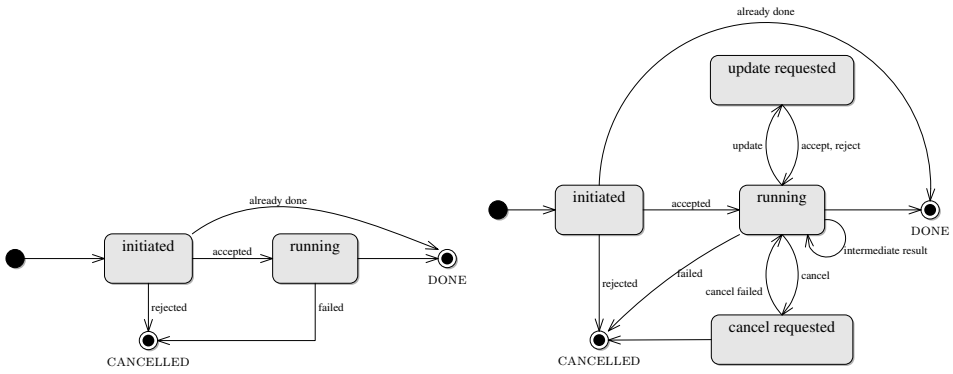
The basic interaction is that the system asks the human for label (e.g. "What is that? [pointing]") and grip information about the objects and then puts them

away. Throughout its interactions, it reports on what it is doing (e.g. "I am now going to pick up the apple"). This simple script leads to an intricate interaction mainly because of two issues: Firstly, the human can interject at any point, to stop or restart tasks, inquire about the information the system has, add or remove objects, and so on. Secondly, speech recognition is limited and clarification dialogs can occur at any point. Full details on the system are available in [11].

The system supports a number of "basic" tasks, which are those that are directly implemented and can merely be configured. "Compound" tasks, are created by sequencing a particular configuration of basic tasks. Example basic tasks include text-to-speech output, moving hand and arm and gazing at something. Example compound tasks include asking for an object label while pointing and looking at the object, as well as picking up an object and putting it away while explaining each step, so that the user stays informed.

3 The Task-State Coordination Pattern

The central idea of the Task-State pattern is to separate the *state* from the *configuration* of a task. The state, then, is described by a finite-state machine (FSM) that is the same for all tasks. While the FSM may in principle by arbitrarily complex, we have found that the one depicted in figure 2(a) is the simplest useful FSM, whereas the one in 2(b) supports all the distinctions we have needed for several complex systems.



(a) Basic Task State Machine (b) General Task State Machine

Fig. 2. Example Task-State Machines

For coordination, only changes in state are relevant. The Task-State pattern requires, firstly, that each such change causes an event notification, and secondly, that the event notification includes the current configuration. Individual components may thus use the configuration easily.

3.1 Example of Use

To see how the event notifications are applied to coordinate task execution, figure 3 presents a full example. In it, each arrow represents an event notification and the first word of the notification label refers to the new task state.

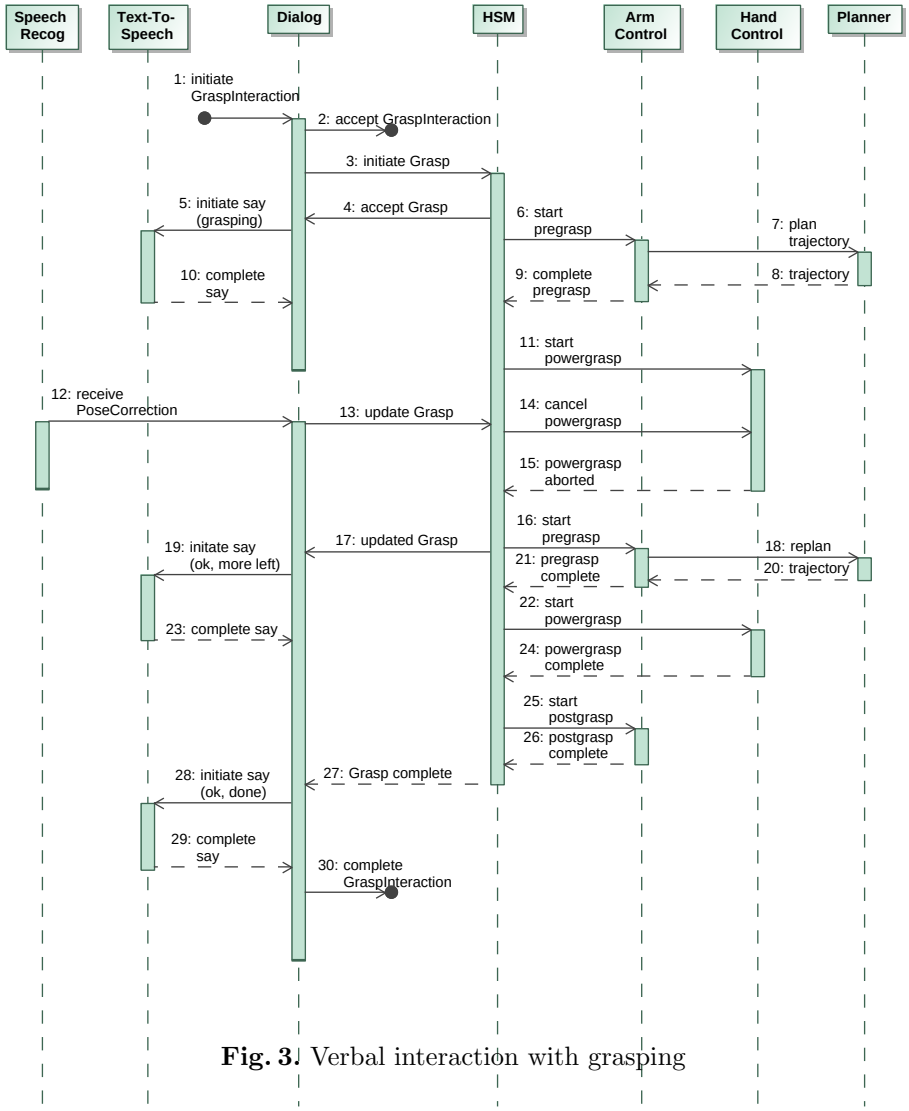


Fig. 3. Verbal interaction with grasping

In the example, we would like to emphasize two main points. Firstly, there are basically two independent sub-systems present: The interaction sub-system (speech recognition, text-to-speech, dialog) on the left-hand side, and motor control on the right hand side. Within each sub-system, tasks are only initiated

and completed or aborted. Between the sub-systems, however, there are also "update" notifications. The reason for this addition is so that the context of these sub-system can be preserved during coordination.

Secondly, please note the close interaction *within* the sub-systems, coordinated by a much simpler interaction *between* sub-systems. This results in loose coupling between sub-systems, an architecturally desirable property. Figure 4, which depicts the information relationships, also visualizes this fact.

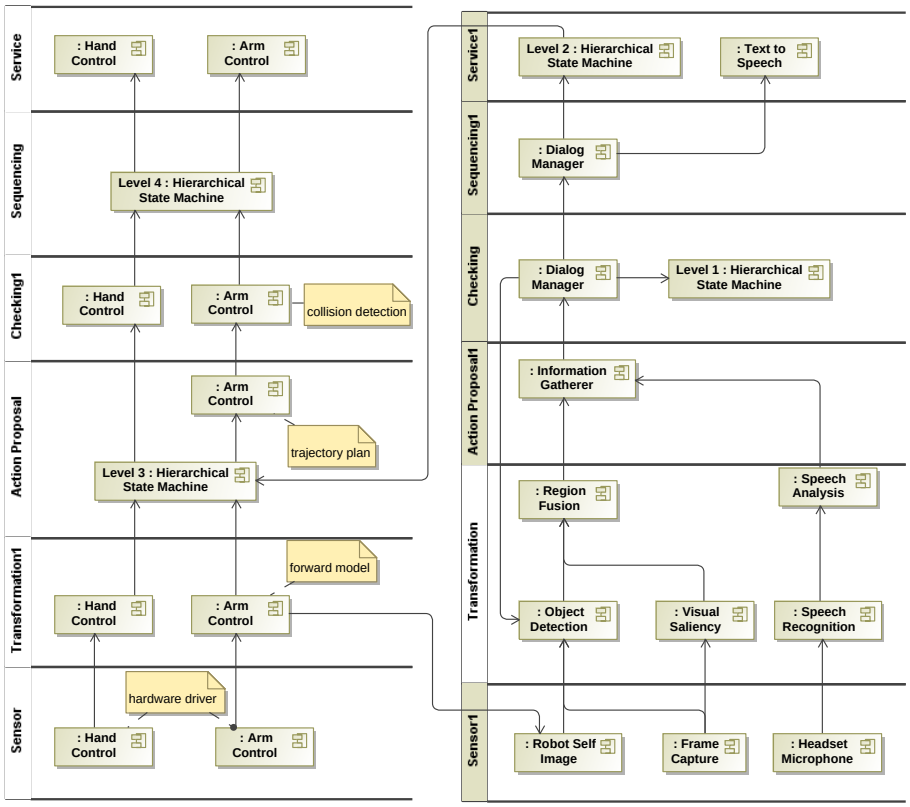


Fig. 4. Sub-system interaction. Each column represents a single sub-system. Only two links exists between them, keeping coupling to a minimum.

4 Conclusions

We have shortly summarized the Task-State pattern, a general coordination mechanism focused on a separation of concerns to facilitate integration. A paper on the dialog manager of the system will be part of the proceedings and more details on the pattern in general are currently in press [12].

Bibliography

[1] Parnas, D.L.: On the criteria to be used in decomposing systems into modules. Commun. ACM **15** (1972) 1053–1058

[2] Utz, H., Sablatnog, S., Enderle, S., Kraetzschmar, G.: Miro - middleware for mobile robot applications. Robotics and Automation, IEEE Transactions on **18** (2002) 493–497

[3] Metta, G., Fitzpatrick, P., Natale, L.: Yarp: Yet another robot platform. International Journal of Advanced Robotic Systems **3** (2006) 43–48

[4] Stewart, D.B., Khosla, P.K.: Rapid development of robotic applications using component-based real-time software. Intelligent Robots and Systems, IEEE/RSJ International Conference on **1** (1995) 465+

[5] Brugali, D., Brooks, A., Cowley, A., Côté, C., Domínguez-Brito, A., Létourneau, D., Michaud, F., Schlegel, C.: Trends in component-based robotics. In Brugali, D., ed.: Software Engineering for Experimental Robotics. Volume 30 of Springer Tracts in Advanced Robotics. Springer Berlin Heidelberg, Berlin, Heidelberg (2007) 135–142

[6] Lefebvre, D.R., Saridis, G.N.: A computer architecture for intelligent machines. In: Proceedings of IEEE International Conference on Robotics and Automation. Volume 3. (1992) 2745–2750

[7] Chatila, R.: Deliberation and reactivity in autonomous mobile robots. Robotics and Autonomous Systems **16** (1995) 197–211

[8] Simmons, R., Apfelbaum, D.: A task description language for robot control. In: Proc. of Conference on Intelligent Robotics and Systems. (1998)

[9] Wrede, S., Hanheide, M., Wachsmuth, S., Sagerer, G.: Integration and coordination in a cognitive vision system. In: International Conference on Computer Vision Systems (ICVS), St. Johns University, Manhattan, New York City, USA, IEEE (2006)

[10] Hanheide, M., Sagerer, G.: Active memory-based interaction strategies for learning-enabling behaviors. In: International Symposium on Robot and Human Interactive Communication (RO-MAN), Munich (2008)

[11] Lütkebohle, I., Peltason, J., Schillingmann, L., Elbrechter, C., Wrede, B., Wachsmuth, S., Haschke, R.: The Curious Robot - Structuring Interactive Robot Learning. In: International Conference on Robotics and Automation, Kobe, Japan, Robotics and Automation Society, IEEE (2009)

[12] Lütkebohle, I., Philippsen, R., Pradeep, V., Marder-Eppstein, E., Wachsmuth, S.: Coordination and Control for Complex Robot Software Systems: The Task-State Pattern. Journal of Software Engineering for Robotics (in press) submitted.