Information Assistance for Smart Assembly Stations

Mario Aehnelt[1] and Sebastian Bader[2]
[1]*Fraunhofer IGD, Joachim-Jungius-Str. 11, 18059 Rostock, Germany*
[2]*MMIS, University of Rostock, Albert-Einstein-Str. 22, 18059 Rostock, Germany*

Keywords: Smart Manufacturing, Information Assistance, Cognitive Architectures.

Abstract: Information assistance helps in many application domains to structure, guide and control human work processes. However, it lacks a formalisation and automated processing of background knowledge which vice versa is required to provide ad-hoc assistance. In this paper, we describe our conceptual and technical work to include contextual background knowledge in raising awareness, guiding, and monitoring the assembly worker. We present cognitive architectures as missing link between highly sophisticated manufacturing data systems and implicitly available contextual knowledge on work procedures and concepts of the work domain. Our work is illustrated with examples in SWI-Prolog and the Soar cognitive architecture.

1 INTRODUCTION

Evaluations show that people with a detailed work plan complete their tasks faster than without it, even if they did not carry out the planning themselves (Kokkalis et al., 2013). This is a key motivation for intelligent systems which assist the worker by creating work plans autonomously and guide him through single work procedures aiming to improve both efficiency and effectiveness of his work. Such intelligent systems will help in manufacturing to ensure a high product quality even when working with insufficiently qualified personnel. They inform the worker about current and upcoming tasks, provide him with detailed knowledge on assembly procedures, or monitor correct work order execution.

Although manufacturing industries already use powerful data management systems that support the planning, execution and monitoring of production processes, there is still a lack of methods and technologies which bring intelligent assistance to the shop floor. Basically, it lacks an automated processing of the lion's share of domain dependent background knowledge. It is hidden in work related standards, regulations, guidelines or simply maintained by experts, thus not available for the average worker.

Our research specifically addresses information assistance for assembly stations at the manufacturing shop floor. Although *smart factories* establish digitalisation and automation to streamline manufacturing processes and quality, there is still the need for manual assembly operations (Würtz and Kölmel, 2012). Here, it requires systematic information assistance in order to manage the complexity and heterogeneity of extremely small lot sizes.

Throughout this paper we focus on smart factories in which individually customised products are assembled by humans. In particular, we assume a lot size of one. This implies, that basically every product is unique and required new construction plans. In contrast to larger lot sizes or series production, it is not profitable to invest much into product-specific assistance systems. Therefore, we need to derive useful assistance systems from existing information sources.

This paper is organised as follows: First, we discuss different types of assistance within the focus of manual assembly processes. Sec. 3 and 4 describe how to detect important situations and how to provide assistance, respectively. In Sec. 5 we show how cognitive architectures can be used to formalize and process the missing background knowledge. We conclude our work with Sec. 6.

2 REQUIRED ASSISTANCE

Below we discuss use cases in which different types of assistance are required to support manual assembly processes. But first we give a short introduction to assembly work activities.

The assembly of machines and technical systems is an essential part of production. It consumes up to

Aehnelt M. and Bader S..
Information Assistance for Smart Assembly Stations.
DOI: 10.5220/0005216501430150
In *Proceedings of the International Conference on Agents and Artificial Intelligence* (ICAART-2015), pages 143-150
ISBN: 978-989-758-074-1

40 percent of costs and even 70 percent of production time. During an assembly single machine parts are joined to first-order assemblies (*pre-assembly*), then to assemblies of higher order (*intermediate assembly*) and finally to an end product (*final assembly*). The German VDI guideline 2860 (German Engineers' Association, 1990) further differentiates between *primary assembly*, which includes the main joining operations as defined with the DIN 8580 (DIN Deutsches Institut für Normung e.V., 2003), and *secondary assembly*, including all additional assembly activities like handling, adjustment and control of parts, material, tools, and machines. Assembly operations such as joining directly refer to physical activities of the worker. A more detailed description of the manual operations as well as of the physical work environment can , for example, be found in (Aehnelt et al., 2014).

Traditionally, a worker is equipped with work orders describing the work to be done on an abstract level. It strongly depends on his individual knowledge and experiences to interpret the given information correctly. Information assistance, especially in complex and continuously changing assembly work processes, improves the quality of work by ensuring an immediate transfer of required work information to the workplace and back to planning systems for example. Here, we differentiate between five general types of information assistance:

- *Raising awareness*: The worker requires up-to-the-minute knowledge about his direct and relevant work environment in order to align his own activities accordingly. Knowing early enough the malfunction or breakdown of a machine, which produces parts for his own assembly, influences his situational decisions and activities. Thus, information assistance needs to make the worker aware of relevant states, events and occurrences within the work environment which have an influence on the planning and execution of the worker's tasks.
- *Guiding*: The worker requires orientation with respect to his current and upcoming assembly tasks. This needs to be given through operational guidance which filters available information for each specific work step, in order to reduce the parallel information load to a required minimum. Knowing the exact joining procedure beforehand does not reduce the risk of failures, especially in complex assembly cases. Thus, information assistance needs to split complex procedures into smaller but easier understandable parts, used to guide the worker step-by-step through the assembly.
- *Monitoring*: In the first place, monitoring the assembly process has a practical value. It allows a detailed comparison as well as re-calculation of planned and real time figures. Additionally, it enables the early identification of quality issues or interruptions. Information assistance needs to collect required data from the workplace which supports the continuously production re-planning. It also needs to control the correct execution of assembly procedures to avoid reworking in case of wrong assembly orders, skipped parts or incorrect tool usage.
- *Documenting*: When it comes to quality issues or even complaints, information assistance needs to support tracking back these issues to their roots, which requires a parallel documentation of assembly tasks. However, this kind of documentation can also be helpful to evaluate assembly procedures finding examples of best practice or expert knowledge inherited within individual work processes.
- *Guarding*: The physical and cognitive loads at the assembly workplace vary from situation to situation. Information assistance needs to guard the worker from overload by balancing the load levels within healthy borders or by visualising it.

Based on the use cases introduced above, we discuss the current state of the art in assistance for manual assembly tasks. Smart assistance is no novelty in manufacturing. However, we find there a majority of specialised and single task solutions focussing on quality assurance and information transfer (Berndt and Sauer, 2012). Other research deals with concepts for smart factories which automate the planning and execution of manufacturing processes in autonomously working factories. Focussing on the automation of robot cells Mayer et al. proposed the usage of intelligent systems to resemble human decision making and problem solving for complex assembly tasks (Mayer et al., 2011). They introduce the *cognitive control unit* (CCU) which ensures the numerical planning of robot behaviour backed by a cognitive architecture. Cognitive architectures can be understood as a mean to implement intelligent and autonomous behaviour in assistance applications. They have proven capable of supporting complex problem solving tasks. A recent example is the simulation of mission management for unmanned aircrafts (Gunetti et al., 2013).

Our own work contributes to the growing demand of information assistance for manual work operations in manufacturing which underlies human flexibility and failures as motivated by (Bader and Aehnelt, 2014). In particular, we focus on assistance that can be generated automatically from existing knowledge

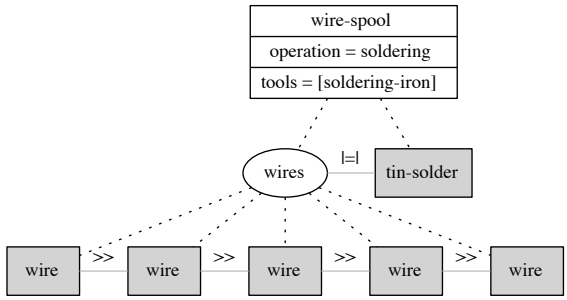

Figure 1: A simple task model stating that a wire-spool is build from 5 wires and tin-solder. In addition, the top-most node contains the joining operation (*soldering*) and the tools to be used (*soldering-iron*).

sources. This allows to use the approach also for small lot sizes.

3 DETECTING SITUATIONS

To actually provide assistance, we first need to recognize situations in which this is necessary. We will discuss the recognition process only briefly, because we simply utilize an approach presented at ICAART 2014. Therefore, we only review some ideas presented in (Bader and Aehnelt, 2014). Based on a formalization of assembly tasks, using so-called task models, a Hidden Markov Model (HMM) is synthesized. The resulting HMM is used to filter sensor data and compute a probability distribution over the current state of the process. As sensory inputs only the processed building blocks are used and the accuracy is analysed with respect to different sensor errors. Here, we use a similar approach, but instead of using the single building blocks as sensory inputs, we also utilize the tools used for the assembly. Figure 1 shows such an enhanced task model, stating that a *wire-spool* is built from 5 *wires* and some *tin-solder*. The parts are joined by *soldering* them using a *soldering-iron*.

To raise awareness for important state changes in the environment as well as for guiding the worker, the current state of the assembly process needs to be determined. For this, we rely on the results reported in (Bader and Aehnelt, 2014), indicating that this is indeed feasible. Based on our annotated models, we believe that the results should be even better, but a detailed analysis is subject to future work.

To detect assembly errors or deviations from the usual procedure is harder to track. In (Bader and Aehnelt, 2014), the authors describe how the system reacts with respect to different types of sensor errors (missing and repeated readings).

But here, we need to detect actual assembly errors, which has not been tackled before. As the task model describes valid construction paths only, the resulting HMM will always compute a probability distribution over valid paths. Therefore, an assembly error is not directly observable. To track assembly errors nonetheless, we analyse the development of the probability distribution over states. Because valid paths result in rather crisp probability distributions (i.e., one path has a very high probability while all others a low one), errors result in a higher entropy of the distribution. Based on this insight, we are able to detect situations in which an error occurred.

```
produces('WireMachine', wire).
task(wire−spool,          % goal
  soldering,              % operation
  [soldering−iron],       % tools
  [tin−solder, 5∗wire]).  % parts
```

Listing 1: Static background information describing that a WireMachine produces wires and that a wire-spool is assembled by soldering wires using a soldering-iron. The task corresponds to the task-model shown in Figure 1.

```
storage(wire, 3).
assembles(worker1, wire−spool).
broken('WireMachine', 'no copper available').
knows(worker1, howTo(soldering)).
```

Listing 2: Dynamic background information describing that 3 wires are available, worker1 assembles a wire-spool, and that the wire machine is broken.

4 PROVIDING ASSISTANCE

Information assistance at the assembly workplace aims at the continuous information exchange between leading manufacturing data systems (enterprise resource planning, manufacturing execution, etc.) and the worker in order to support efficient working and to avoid interruptions, failures or quality related issues.

To keep argumentation and presentation simple, we assume that the state of the world is known exactly. The system described here, has been implemented in SWI-Prolog[1]. The state of the world is described as static and dynamic facts shown in Listing 1 and 2, respectively.

Below, we work out information which helps to provide awareness on the assembly situation, guides the worker through an assembly, and finally monitors his work and results.

[1] http://www.swi-prolog.org

```
informationDemand(InfoType, Person, Info) :-
  % Person assembles a given item ..
  assembles(Person, Item),
  % ... containing a Part with a given Quantity
  isPartOf(Part, Item, Quantity),
  % The Machine producing the Part ...
  produces(Machine, Part),
  % ... is broken for a given Reason
  broken(Machine, Reason),
  % The Quantity is larger than the StoredQuantity
  storage(Part, SQ), Quantity > SQ,

  InfoType = awareness(stateOf(Machine)),
  Info = brokenMachine(Machine, Part, SQ).
```

Listing 3: Specification of an information demand to raise awareness with respect to a broken machine. The predicate `isPartOf` allows to access sup-parts as defined through the task model.

4.1 Raising Awareness

Situational awareness with respect to his work environment helps the worker to orientate within complex processes and to align his own activities accordingly (Gutwin and Greenberg, 2002). It can be understood as inherent information demand of carrying out and completing work tasks. All changes of the virtual or physical work environment which influence the ongoing or upcoming assembly tasks need to made aware to the worker. This includes, for example:

- *planned tasks* which can change in time and priority,
- *missing material, tool or information* which are required but not available for assembly, or
- *deviations* from normal procedures, orders and qualities.

Depending on the information impact as well as urgency, assistance needs to help perceiving it embedded in the ongoing work flow.

After evaluating the information demand as specified in Listing 3 with respect to the current state of the world, InfoType is unified with awareness(stateOf(*'WireMachine'*)) and Info with brokenMachine(*'WireMachine'*, wire, 3). In addition, all premisses are known. This allows to generate the output shown in Figure 2. A simple verbalisation engine is used to translate the predicates (e.g., shown in bold font in Listing 3) instantiated while computing the information demand into english sentences. Predicates corresponding to dynamic facts are verbalised as assumptions.

Type: *awareness(stateOf(WireMachine))*
User: *worker1*
Info: *We are running low on wire, because* WireMachine *is broken and only 3 items are left in storage.*
Explanation: *It is assumed that* worker1 *assembles* wire-spool. *Item* wire *is needed* 5 *times to assemble a* wire-spool. WireMachine *produces* wire. *It is assumed that* WireMachine *is broken, because* no copper available. *It is assumed that* 3 *items of* wire *are left in storage.* Worker1 *should be aware of the state of* WireMachine.

Figure 2: Infomessage and explanation generated for the information demand specified in Listing 3.

4.2 Guiding

Similar to formal education processes, information assistance in form of guiding can be understood as an informal way of mediating and learning facts (*what*), procedures (*how*) and concepts (*why*) required for a specific assembly task. The shaping and depth of guidance varies depending on a specific assistance objective to be supported (Aehnelt and Urban, 2014). In a first step the worker is required to *remember*, *understand* and *apply* the given information in order to prepare and execute his assembly task correctly. Thus, information assistance has to collect and visualise:

- *bills of material* which identify the material to be used for an assembly step,
- *bills of tools* which lists the tools and machines to be used for joining procedures,
- *procedures* which describe the correct handling, adjusting, joining, and controlling of materials and tools including safety relevant information, and
- *planned figures* which detail the expected assembly times and results.

As motivated in Section 2, it is important to split complex assembly procedures into smaller instructions (*steps*), reducing thus cognitive loads for the worker and giving them a clear work structure (Kokkalis et al., 2013). Showing then the required information parallel to the ongoing work process for each step only, helps to achieve the three assistance objectives.

Similar to the information demand specified in Listing 3, it is possible to formalise guiding knowledge. This includes for example the tool to be used for the current assembly operation as shown in Listing 4. Figure 3 shows the resulting output after verbalising the predicates.

```
informationDemand(InfoType, Person, Info) :-
  % Person assembles a item
  assembles(Person, Item),
  % get Join-Operation from task model
  task(Item, JoinOp, Tools, Parts),

  InfoType = guiding(JoinOp),
  Info = currJoinOp(Person, Item, JoinOp, Tools).
```

Listing 4: Specification of an information demand with respect to the current type of join operation and the tools to be used.

Type: *guiding(JoinOp)*
User: *worker1*
Info: *You should* soldering *the* wire-spool *using* soldering-iron.
Explanation: *It is assumed that* worker1 *assembles* wire-spool. *Item* wire-spool *is assembled via* soldering, *using a* soldering-iron.

Figure 3: Infomessage and explanation generated for the information demand specified in Listing 4.

Similar specifications can be used to refer to the items to be used. For complex assembly tasks, many different types of joining operations have to be performed by the worker. Usually, not every worker has the same in-depth training for all of them. Therefore, more background knowledge should be provided for unknown operations. This can be formalised as shown in Listing 5.

```
informationDemand(InfoType, Person, Info) :-
  % Person assembles a given part
  assembles(Person, Part),
  % get information from task model
  task(Part, JoinOp, Tools, _Children),
  % if there is a description D of the op ...
  bgInfo(joinOp(JoinOp), description(D)),
  % ... and the person does not know it
  not( knows(Person, howTo(JoinOp)) ),

  InfoType = bgInfo(joinOp(JoinOp))
  Info = bgInfo(joinOp(JoinOp), description(D)).
```

Listing 5: Specification of an information demand with respect to missing background knowledge.

Assuming knows(worker1, howTo(soldering)) to be true, and false for worker2, results in different information assistance for both.

4.3 Monitoring

Monitoring the ongoing assembly activities of the worker establishes a feedback channel to the information assistance system. It requires a close observation of work progress, rejects, and issues for practical reasons. Manufacturing execution systems need this information to allow a detailed planning of production processes. For situation recognition we require more detailed knowledge from monitoring: the *material* taken as well as *tools* picked up and their *configuration*, which enables us to draw conclusions on the current assembly step executed and possible deviations in comparison to the provided instructions.

5 USING SOAR TO PROVIDE ASSISTANCE

The previous sections showed conceptual and technical considerations with respect to acquiring knowledge on actual assembly situations as well as to providing information assistance at the assembly workplace. Although, a vast amount of required information can already be found in manufacturing data management systems, the major share of procedural and conceptual knowledge is not yet formalised in systems which allow their automated processing (see Figure 4). We find work instructions, standards, or assembly guidelines normally written in natural language within accompanying documents. However, there has already been research to distinguish between the semantic meaning of instructions and their visual representation (Mader and Urban, 2010) based on controlled vocabularies which allows for automation. They still require a manual authoring of instructions for each assembly process individually. What we require in contrast for guiding for example, is an abstract formalisation of assembly procedures in general which is filled at runtime with factual knowledge about the specific product or situation.

Cognitive architectures are a mean to bridge the gap between common assembly descriptions, that we find in VDI 2860 or DIN 8580, and reusable procedural knowledge in information assistance systems. Below, we summarize our approach of utilising the cognitive architecture *Soar* for assisting the worker during assembly.

5.1 Cognitive Architectures

Cognitive architectures can be traced back to Newell's early hypothesis that any artificial intelligence is based on a symbol system and related rules (Newell, 1980). As of today, a cognitive architecture describes the *mental structure* for human information processing, the *representation* and *organization* of information within these structures as well as the *functional*

processing required to acquire, use, and modify information (Langley et al., 2008). Hence, they allow modelling and implementing intelligent behaviour in smart applications and environments. We also need it for providing assistance as described in Section 2. In our own work we use the cognitive architecture Soar (*state, operator and result*) for:

- *situation detection* based on observations of the physical work environment and following reasoning,
- the *formalisation and processing* of contextual background knowledge (e.g. procedures, explanations),
- *interactions* with the worker as well as the physical environment to provide assistance by raising awareness and guiding for example, and for
- additionally *learning* assembly related practices from observation.

In general, processes in Soar are related to the gradual alternation of information and states in working or long-term memory (Laird, 2012). Here, a situation is formalized as a state in working memory, which is modified by evaluating and applying *operators* until an intended final state is reached. The operator definition consists of required conditions and actions on the working memory. It inherits procedural and conceptual knowledge from the corresponding knowledge domain. New operators can also be derived by observation of decision making processes (*chunking*) and through learning processes.

First examples on how we use Soar to provide assembly assistance are illustrated below.

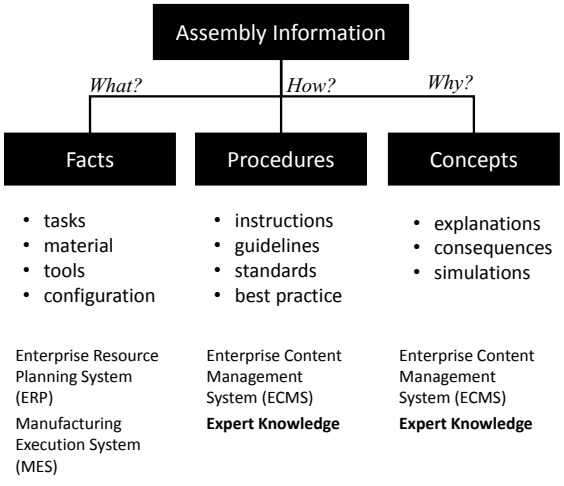

Figure 4: Required information is contained in different enterprise resources. Partly it is individual expert knowledge which is not externalised in any management system.

5.2 Situation Detection

In Soar we represent the individual situations of the work environment during an assembly by *states*. Each state identifies a different condition of elements within the Soar working memory, which finally holds a virtual copy of the physical environment. Thus, we transfer the state of real objects, e.g. tools, the material stack, or even the workplace, into logical *objects* and their *attributes* in working memory. Soar connects then to sensors which allow the observation of individual object states and events, e.g. the usage of a tool, in order to update attributes of the related working memory object. The working memory is so the basis of all reasoning on discrete states of the work environment.

In Listing 6 we illustrate the usage of Soar's state operator mechanism for a specific soldering situation. It consists of two parts, an operator proposal rule and an application rule. The first one defines the precondition of a state described by working memory objects and their attributes. In our example it requires at least two materials or parts and a soldering iron in order to start a soldering operation. If the current state of the working memory matches these conditions, the operator *join-part* becomes candidate in following decision making.

```
# parts can only be soldered if there are at least
# two parts taken, a soldering iron and solder to
# support the joining operation
sp {propose*soldering-parts
   (state <s> ^name assemble)
   (<s> ^count_taken > 1)
   (<s> ^iron_taken <=> yes)
   (<s> ^solder_taken <=> yes)
-->
   (<s> ^operator <o> + =)
   (<o> ^name soldering-parts )                    }

# soldering parts reduces the taken parts to a
# single compound part and consumes solder
sp {apply*soldering-parts
   (state <s> ^operator.name soldering-parts)
   (<s> ^count_taken <t>)
   (<s> ^solder_taken <m>)
-->
   (<s> ^name soldering)
   (<s> ^count_taken 1)
   (<s> ^count_taken <t> -)
   (<s> ^solder_taken <m> -)                       }
```

Listing 6: Definition of operator and production rules in Soar for joining assembly step.

Soar evaluates the likelihood of each operator based on the current state of objects in working memory and all candidate proposal rule definitions. It uses

contextual knowledge (see Section 5.3) to compare and select candidate operators during decision making.

With applying finally the operator *soldering-parts* the working memory state is changed by modifying single memory objects, e.g. by reducing the amount of available parts.

5.3 Contextual Knowledge

One of Soar's strengths lays in its interaction between working and long-term memory. While the working memory holds information about the current condition of logical objects (see Section 5.2), the long-term memory represents the contextual knowledge which is required to select and apply an operator. Here, we find five different knowledge types in Soar: knowledge which qualifies an operator for a situation, knowledge to compare operators, knowledge to select a single one, knowledge to change the working memory, and knowledge to elaborate a state. All types contain contextual knowledge of the application work domain, in our case of assembly activities as formulated in VDI 2860 and DIN 8580.

In Listing 7 we encode this assembly knowledge to define the sequential order of single work steps and their requirements. In this example, we require further solder material prior to collecting the soldering iron, which will be defined by additional preference operators, such as +,>,<, or *!*.

```
# soldering parts requires additional solder
# material which need to be taken first
sp {propose*take-solder
   (state <s> ^name assemble)
   (<s> ^iron_taken <=> yes -^solder_taken)
-->
   (<s> ^operator <o1> + ; <o1> > <o2>)
   (<o1> ^name take-solder)
   (<o2> ^name take-iron)                    }
```

Listing 7: Contextual knowledge of the work domain is encoded in Soar's production rules.

In this way, we were able to transfer the relevant assembly process logic into Soar operators. They help us guiding the worker with small sized assembly instructions as required for an automated information assistance.

5.4 Interaction

We also use Soar to establish an interaction between information assistance system and the worker as well as vice versa. As described in Section 4 we aim to raise the worker's awareness with respect to relevant information on his ongoing assembly process, guide him step by step through the assembly, and monitor his work activities. It finally requires the interaction to inform him and collect data from him. This can also be formalised by operator rules.

```
# provide information assistance once the worker
# is not informed on his following work step
sp {propose*inform
   (state <s> ^type state)
   (<s> ^name <n> - ^is_informed)
-->
   (<s> ^operator <o1> !)
   (<o1> ^name inform)                       }
```

Listing 8: Information assistance is modeled as operators in Soar.

Listing 8 shows the proposal rule of an *inform* operator which will provide the worker with instructions for his next assembly steps. In a similar manner we define operators for raising awareness for example.

6 CONCLUSIONS

In this paper we showed how to provide information assistance for a smart assembly station. After defining different types of information demands, we briefly discussed a possibility to detect situations in which assistance is needed. Then we showed how assistance can be provided using a crisp state of the world and logical specifications of the information demand (using an implementation in SWI Prolog). Even though, most information can already be found in manufacturing data management systems, the majority of procedural and conceptual knowledge is not yet formalised. To bridge this gap we use the Soar architecture.

Although the concept of cognitive architectures is not new to implementing systems with intelligent behaviour, it is still rarely used to make the contextual background knowledge from an application domain accessible for complex problem solving tasks. Our approach shows on both, conceptual as well as technical level, the usage of an cognitive architecture for supporting information assistance on the manufacturing shop floor. It illustrates the role of logical modelling and the transfer of implicit and barely formalised knowledge into predicate logic and state operators.

However, one of the next required steps is to learn new procedures and novel connections from observation of real assembly activities. Here it is promising to start with a basic guidance skeleton and detail the missing assembly steps by tracking, interpreting, and learning from work procedures of assembly experts.

In addition, we are working on an experimental evaluation of the ideas. This includes the collection of real sensor data and the formalisation of real-world examples. And finally we will work on an automated transfer of real existing knowledge into an assistance system.

REFERENCES

Aehnelt, M., Gutzeit, E., and Urban, B. (2014). Using activity recognition for the tracking of assembly processes: Challenges and requirements. In Bieber, G., Aehnelt, M., and Urban, B., editors, *WOAR 2014*, pages 12–21. Fraunhofer-Verlag, Stuttgart.

Aehnelt, M. and Urban, B. (2014). Follow-me: Smartwatch assistance on the shop floor. In *Proceedings of the 16th International Conference on Human-Computer Interaction (HCII)*, Lecture Notes in Computer Science. Springer-Verlag.

Bader, S. and Aehnelt, M. (2014). Tracking assembly processes and providing assistance in smart factories. In *Proceedings of the 6th International Conference on Agents and Artificial Intelligence (ICAART)*.

Berndt, D. and Sauer, S. (2012). Visuelle assistenzsysteme in der montage verhindern ausfälle. *MM Maschinen-Markt*, (19):46–49.

DIN Deutsches Institut für Normung e.V. (2003). Din 8580 manufacturing processes - terms and definitions, division.

German Engineers' Association (1990). Vdi 2860:1990-05 assembly and handling; handling functions, handling units; terminology, definitions and symbols.

Gunetti, P., Dodd, T., and Thompson, H. (2013). Simulation of a soar-based autonomous mission management system for unmanned aircraft. *Journal of Aerospace Information Systems*, 10(2):53–70.

Gutwin, C. and Greenberg, S. (2002). A descriptive framework of workspace awareness for real-time groupware. *Computer Supported Cooperative Work (CSCW)*, 11(3):411–446.

Kokkalis, N., Köhn, T., Huebner, J., Lee, M., Schulze, F., and Klemmer, S. R. (2013). Taskgenies: Automatically providing action plans helps people complete tasks. *ACM Transactions on Computer-Human Interaction*, 20(5):1–25.

Laird, J. E. (2012). The soar cognitive architecture. *Artificial Intelligence and Simulation of Behavior Quarterly*, (134):1–4.

Langley, P., Laird, J. E., and Rogers, S. (2008). Cognitive architectures: Research issues and challenges. *Cognitive Systems Research*, 10(2):141–160.

Mader, S. and Urban, B. (2010). Creating instructional content for augmented reality based on controlled natural language concepts. In *Proceedings of 20th International Conference on Artificial Reality and Telexistence (ICAT 2010)*.

Mayer, M. P., Odenthal, B., Wagels, C., Kuz, S., Kausch, B., and Schlick, C. M. (2011). Cognitive engineering of automated assembly processes. In Harris, D., editor, *Engineering Psychology and Cognitive Ergonomics*, volume 6781 of *Lecture Notes in Computer Science*, pages 313–321. Springer Berlin Heidelberg, Berlin, Heidelberg.

Newell, A. (1980). Physical symbol systems*. *Cognitive Science*, 4(2):135–183.

Würtz, G. and Kölmel, B. (2012). Integrated engineering – a sme-suitable model for business and information systems engineering (bise) towards the smart factory. In Camarinha-Matos, L., Xu, L., and Afsarmanesh, H., editors, *Collaborative Networks in the Internet of Services*, volume 380 of *IFIP Advances in Information and Communication Technology*, pages 494–502. Springer Berlin Heidelberg.