Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 41 (2016) 224 – 229

www.elsevier.com/locate/procedia

48th CIRP Conference on MANUFACTURING SYSTEMS - CIRP CMS 2015

Interactive planning of manual assembly operations: from language to motion

Stephan Busemann[a]*, Jörg Steffen[a], Erik Herrmann[a]

[a]*Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI) GmbH*
Stuhlsatzenhausweg 3, D-66133 Saarbrücken, Germany

* Corresponding author. +49-681-85775-5286. *E-mail address:* stephan.busemann@dfki.de

Abstract

This paper describes part of a novel view of planning the assembly of cars at the shop floor, which is currently being explored in the EU-funded project INTERACT, in which 3D worker simulations are automatically generated from textual descriptions. Under this view, all planning is carried out virtually, thereby interactively exploiting the workers' knowledge.
We suggest solutions to the subtask of mapping textual descriptions onto motion sequences. Consider a description like "Tighten arm support with cordless screw driver on center console". This text belongs to a controlled natural language that is – different from unconstrained language – amenable to unambiguous linguistic analysis. The result is then broken down into a sequence of elementary actions, such as WALK, PICK, or PLACE, carried out by a digital human model (DHM) using and manipulating objects in the 3D scene. The representation level of elementary actions is designed to provide all information needed for subsequent motion synthesis. For proper 3D visualization, each action requires dynamic or static parameters such as the grasp points at the objects, and positions of the DHM and the objects.
A semantic interpretation of the linguistic results must account for all variations to be expected in the scene. For instance, if the DHM is not "near" the object of interest, it must WALK. Evaluating this kind of condition-action rules against constraints imposed by the scene for a given textual description leads to a sequence of elementary actions that is then processed further and visualized. When viewing the simulation, the human planner can affect some aspects of processing, such as the order of elementary actions, by causing manipulations to the rules.

Keywords: Planning; Knowedge based system; Simulation; Human Language

1. Introduction

The assembly of cars at the shop-floor can be described by a sequence of detailed human language task descriptions. The worker is supposed to carry out each task accordingly at the respective work station, using skills acquired by instruction and experience. For new product types, new task descriptions are needed, and the ways in which the tasks should be carried out are studied and defined before production starts. This planning process is currently carried out manually by a group of experts discussing the optimum execution of tasks with the help of prototypes.

Using a realistic visual simulation of the assembly operations during planning would allow a more complete and explicit definition of the task, as the requirements, the knowledge, the skills and much of the experience of the experts will be reflected in the visual simulation. Alternative virtual task executions can be generated on the fly. Such a simulation of the assembly process is expected to save time and costs. The pre-production planning workshops will make use of the simulations, aiming at defining a final sequence of task description and, for each task, a complete definition of its execution. This process will be supported by the virtual simulation, which can be manipulated by the experts to improve task execution models.

For the simulation to be realistic, a model of the shop-floor scenario will be required that includes, among many other things, the types and locations of objects, as well as ways to handle them (grasp points).

This innovative vision is currently taking shape through the ongoing EU-funded project INTERACT (cf. http://www.interact-fp7.eu). The present paper focuses on

doi:10.1016/j.procir.2015.12.106

solutions to the initial problem of relating human language task descriptions to a level of annotated elementary actions that can then be processed further yielding motion primitives. This corresponds to the marked left-hand side of Figure 1, which sketches the architecture of components and interfaces used.

The task descriptions used for the simulation are the same as the ones used for manufacturing. Natural language, if used freely, would allow an abundance of statements to express the same task, some more specific than others, some ambiguous without its author even noticing. Therefore a controlled natural language (CNL) is used, which excludes these drawbacks (Section 2.1). This guarantees unambiguous, homogeneous wordings for the same tasks across shop floors and yet preserves the level of human language, which is best suited to have planners describe the tasks.

(1) Tighten arm support with cordless screw driver 3x on center console.

CNL task descriptions such as (1), however, are too course-grained to be mapped directly onto prerecorded motion elements. They are underspecified with respect to the scenario in which the action is to be carried out, and this is on purpose since the task descriptions are meant to generalize over concrete shop-floor scenarios. As an interface layer between task descriptions and motions, a level of elementary actions was defined (Section 2.2) that is complemented with constraints derived from the 3D model of the scenario (Sections 2.3, 2.4). This representation level is fine-grained and specific enough for the Morphable Graphs module (MG++) that synthesizes an animation for a digital human model (DHM) by further breaking down elementary actions into motion primitives.

Mapping course-grained, underspecified linguistic task descriptions onto a sequence of more detailed elementary actions involves adding information. In the case in hand, this information is gathered from the model of the 3D scene (cf. Figure 1). For instance, depending on the respective state of the scenario, the DHM will be made to walk, and to pick, carry and place objects – fine-grained actions never mentioned at the level of task descriptions. In (1) intuition tells that a DHM must get hold of an arm support, a screw driver and three screws and possibly fetch them to the assembly place.

The mapping is carried out in two steps: the creation of CNL-compliant task descriptions (Sections 3.1), and the breakdown onto the level of elementary actions, delivering all information needed for the motion synthesis component (Section 3.2). In the sequel we refer to both these steps jointly as “CNL processing”.

While a first implementation shows the feasibility of the suggested methods, the interaction with the planning experts, who may need to manipulate some aspects of the breakdown, requires additional features that are currently under development (Section 4).

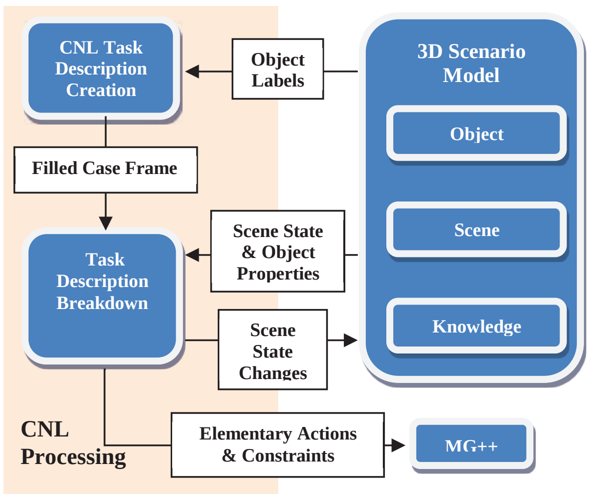
CNL Task Description Creation
Object Labels
3D Scenario Model
Object
Filled Case Frame
Scene State & Object Properties
Scene
Task Description Breakdown
Scene State Changes
Knowledge
CNL Processing
Elementary Actions & Constraints
MG++

Fig. 1. System architecture.

2. Levels of knowledge representation

2.1. Controlled natural language

Controlled natural languages [1-3] are subsets of natural languages such as English. Not all words are permitted, not all grammatical constructions are allowed, and certain stylistic rules are in place to guide the writers.

CNLs were developed to avoid ambiguity in natural language (consider the ambiguous sentence “Sue met Tania because she wanted to apologize.”). In CNLs, the use of pronouns usually is prohibited.

Natural language statements are often unspecific. Consider “The door must be opened before entering” or “The screw driver should be placed near the screws”. In both examples, the actors are not mentioned. In technical writing, specifying the actors is essential. CNLs usually forbid the use of modal verbs and of passive constructions. “Open the door before entering”, or “Place the screwdriver near the screws” are valid expressions in many CNLs.

Natural language statements are often very complex. While parsing free text has made tremendous progress, there are still constructions that cannot be parsed successfully yet. A CNL is defined in such a way that all its expressions can be guaranteed to parse correctly.

For INTERACT, a CNL is needed for all of the above reasons. Planners must be able to express their intentions in a normalized, canonical fashion that can be understood easily by the workers. Task descriptions are short imperative clauses (usually up to 15 words). Assembly operations can be described by an activity that determines a set of semantic roles. For instance, “tighten” requires roles for the object to be tightened, for the target, to which the object is tightened, and for the means of tightening (usually a tool, and/or fasteners). The number of tightening operations can be specified optionally, as in (1) above.

An additional benefit of using the CNL is a unified linguistic terminology across the company since objects und actions are described in a unique manner. Finally, as is generally recognized, using CNLs allows for considerably easier solutions to the notorious problem of multilingual expressions.

2.2. Elementary actions

The level of *elementary actions* was defined as an interface between the CNL breakdown and the motion synthesis components of the INTERACT system. The actions were defined by analyzing what was needed for the pilot cases defined for the project, among them the tasks of center console preassembly and rear light mounting. A total of 22 different elementary actions have been identified, among them actions

- changing the position of objects: INSERT, TURN, WALK, CARRY, MOVE;
- changing the status of object attachment: RELEASE, ATTACH, DETACH;
- changing both: PICK, PLACE;
- interacting with the scene in other ways: TOUCH, USE TOOL.

At the current stage of implementation, the elementary actions WALK, PICK, CARRY and PLACE are supported as these are generic to all pilot cases.

The PICK elementary action describes the process of picking up an object with either one or both hands. PICK has the effect of attaching the object to the corresponding hand joints of the DHM. Objects connected by attachment both change their position whenever one of them is moved. In the case of PICK, the object picked changes its position whenever the DHM moves.

PLACE is the inverse of PICK: The DHM places an object at a specific position, which causes the object to be detached from the corresponding hand joints. It possibly gets attached to a another object, e.g. when placing a tool onto a trolley.

WALK describes a collision-free movement of the DHM from its current to a specified position with empty hands, whereas CARRY does so with the DHM holding an object.

2.3. Constraints

Elementary actions are always accompanied by a set of *constraints*. In INTERACT constraints describe the conditions that an animation must fulfil in order to produce a valid, naturally looking result. Constraints are associated with joints of the DHM skeleton. A joint has a position consisting of x, y and z coordinates in the global coordinate system and an orientation consisting of Euler angles in the x, y and z plane. Constraints on both position and orientation can be underspecified. We differentiate between trajectory constraints and key frame constraints.

Trajectory constraints define a spline for a joint based on a sequence of control points each specifying the position and/or orientation. The trajectory is supposed to be applied as a constraint on the whole elementary action. For instance, a path for WALK or CARRY would be defined as a trajectory constraint for the root joint of the DHM skeleton.

Key frame constraints, on the other hand, define the position and/or orientation of a joint that needs to be reached at a certain key frame. Key frames are manually annotated frames depicting key elements of a motion primitive. As CNL processing has no access to the explicit key frame index in the canonical timeline of a motion primitive, which is part of the motion synthesis component, a key frame constraint refers to a key frame using a predefined identifier that is shared with the MG++ component. For instance, the key frame constraint associated to PICK refers to the key frame “start_contact” and determines the position and orientation of the hand joints when they contact the grip points of the object to be picked. Accordingly, the constraint associated to PLACE refers to the key frame “end_contact” and determines the position and orientation of the hand joints when they release the grip points of the object placed.

Additionally, key frame constraints are annotated in order to reflect changes of the actual scene (as caused by e.g. picking and placing actions), the objects involved, and their mutual attachment status. This is needed since the MG++ module does not have access to the model of the 3D scene.

2.4. Interface to the model of the 3D scene

Obviously the sequence of elementary actions representing a task description depends on the objects and the actual scene the task is supposed to take place in. To make this information available for CNL processing, an interface to the model of the 3D scene has been defined. This interface also allows CNL processing to update the 3D scene with any changes. In the following we describe the main methods of the interface.

Positions and the “nearness” relation between objects are essential for CNL processing. The interface offers methods to retrieve the position for an object and also to query the scene if one object is reachable from the position of another object, the latter one usually being the DHM. If an object to be handled by the DHM is not reachable from its position, a WALK elementary action has to be inserted. In this case, CNL processing needs to retrieve a valid target position for the walk, so that after walking there, the nearness between the DHM and the initial object is given. For that purpose, the interface offers a method to get a collision free path leading the DHM “near” an object.

When handling an object, CNL processing must query the knowledge base of the 3D scene to learn if this requires one or both hands. The interface provides methods to retrieve this information. It determines the constraints of a PICK elementary action to apply. Moreover, it might also be necessary to insert other elementary actions beforehand, e.g. in case the DHM is already holding an object and the next object to handle requires both hands. In this case, the object in hand has to be put down first. The interface offers a method that returns a suitable position for temporarily placing an object.

When creating constraints for PICK and PLACE elementary actions, the grip points of the associated objects must be known. The interface contains methods to retrieve the grip points for the right and left hand of the DHM. Also, in the case of placing an object, the interface contains methods to get the absolute positions of the right and left hand grip points for the target position of the placed object. These are the positions where the hand joints of the DHM have to be moved to, as expressed in the constraints of the PLACE elementary action.

In case of a PLACE elementary action, the target position for the object needs to be retrieved. On the CNL side, the target position is usually described by referring to a target object together with a preposition, e.g. "place on table". The interface contains a method that is able to derive a proper target position.

Whenever an elementary action would cause changes in the 3D scene, the 3D scene model has to be updated accordingly. For that purpose, the interface offers methods to update the position of an object and also to declare objects as attached to each other, or as detached. Whenever the position for an object is changed in the 3D scene, the positions of all attached objects would be adapted accordingly, too.

When referring to objects in the 3D scene, their object identifiers must be known. For that purpose, the interface offers access to a mapping between natural language labels of objects used in the CNL and object identifiers used in the 3D scene model. This is required for creating CNL task descriptions, as described in the following Section 3.1.

3. From language to motion

3.1. Creation of CNL task descriptions

The CNL in INTERACT is characterized by a very limited version of English in which assembly tasks can be expressed. Basically it consists of more or less complex verb phrases (e.g. (1)). Different from the standard use of CNL, which requires the user to learn and apply the CNL "by heart" when writing e.g. technical reports [4], CNL task descriptions in INTERACT can be interactively generated by guiding the user through the available choices. To describe a task in the CNL system, the user selects the activity (tighten) and assigns the individual components (arm support, cordless screw driver, center console) to semantic roles of the activity (theme object, tool, goal) using a menu-based interface.

The user decides about the proper components based on an association of the linguistic description (e.g. "arm support") to all available instances in the scene (e.g. "arm_suppt11"). If there are multiple candidate instances, the user is requested to make a choice.

After selecting an activity, which is expressed by a verb, the process to generate a task description starts with a template consisting of predefined surface strings and placeholders for the roles that remain to be filled (2).

(2) Tighten <NP:THEME> <PP:TOOL> <NUM:1-5>? <PP:GOAL>.

Roles are depicted by a syntactic and a semantic type separated by a colon. The syntactic type describes the syntactic category of a role filler. We distinguish NP (for noun phrases), PP (for prepositional phrases) and NUM (for numbers). Optional roles are marked by a question mark. The semantic type defines the meaning of a role within the activity. There are no more than 10-12 semantic types. The major ones include

- THEME: the part that is handled;
- GOAL: the place towards which the activity happens, e.g. the part with which the THEME part is to be assembled;
- SOURCE: the place from where the activity happens, e.g. the place at which the THEME part is located initially;
- TOOL: the tool with which the activity is executed;
- FASTENER: a fastener to connect parts, e.g. a screw.

Numeric roles define a number interval instead of a semantic category. The interval defines the possible role fillers. Additionally an "x" is appended to the number.

A set of semantic roles is called a case frame [5]. Verbs are assigned to case frames, forming together the required semantics of the sentence. A filled case frame together with the verb and the component identifications suffices to define the linguistic surface structure of the CNL expression.

Only for the PP:SOURCE and PP:GOAL roles, prepositions must be chosen by the user since they may depend on both the verb and the role context in the work task description. PP:GOAL may be complemented by prepositions like "to", "in", "on", "onto" or "at", whereas PP:SOURCE could be used with e.g. "of", "from" or "out of".

By filling all roles, a CNL expression such as (1) with the identified components is generated by the system. At the same time, a filled case frame such as (3) is generated.

(3) [theme: arm_suppt11, tool: cdless_drv2, goal: ccon1, num: 3]

This approach has not only the benefit of being cost-effective, as expensive training of CNL users is obsolete, but also allows for the unambiguous analysis of CNL expressions generated by this system, yielding a filled case frame such as (3) as a result. The next section describes how such case frame representations are broken down into elementary actions.

3.2. Breakdown of task descriptions

The breakdown of a case frame into elementary actions depends on the state of the scenario the action shall be carried out on. In order for the DHM to get hold of the screw driver, the screws and the arm support, various instances of walking, picking, etc. may be needed.

Obviously many parameters influence the generation of a precise sequence of elementary actions to be visualized. Some parameters depend on the objects at stake. For instance, a screw is picked up differently than an arm support (with one vs. both hands). The knowledge about each object that can be grasped by the DHM is depicted in the knowledge base. Other parameters are based on the state of the 3D scenario at the point of time when the action is carried out. They include the positions of the DHM and of the objects at stake.

We assume that the knowledge base and the 3D scene contain information on all objects that may be involved in assembly operations at any time. For each activity, a minimal list of elementary actions is defined that is absolutely necessary to execute the associated activity. Depending on the state of the scene, additional elementary actions are derived during the breakdown and inserted into this list. The minimal list for the pick-and-place activity is PICK and PLACE.

For each elementary action we define a set of preconditions, scene updates and postconditions. For an elementary action to be executed, all of its preconditions must hold in the scene. When executed, the 3D scene is updated. The postconditions state the effects of the executed elementary action that hold in the 3D scene.

Tables 1-4 show the data for the elementary actions discussed above.

Table 1. PICK preconditions, scene updates and postconditions.

Preconditions	Hand(s) to handle object must be empty.
	DHM is near object.
Scene updates	Detach object from all other objects.
	Attach object to hand joint(s) of DHM.
Postconditions	DHM holds the object.

Table 2. PLACE preconditions, scene updates and postconditions.

Preconditions	DHM holds the object.
	DHM must be near the target object where the object is to be placed.
Scene updates	Detach object from hand joint(s) of DHM.
	Attach object to target object.
	Update object's position.
Postconditions	DHM has empty hand(s).

Table 3. WALK preconditions, scene updates and postconditions.

Preconditions	DHM has empty hand(s).
Scene updates	Update position of DHM.
Postconditions	DHM is near target position.

Table 4. CARRY preconditions, scene updates and postconditions.

Preconditions	DHM holds object to be carried.
Scene updates	Update position of DHM.
Postconditions	DHM is near target position.

If the preconditions of an elementary action scheduled to be executed are not met by the current 3D scene – e.g., the DHM is not near an object to be PICKed – one or more additional elementary actions have to be carried out beforehand in such a way that their effects lead to a state of the 3D scene that is compatible with the preconditions of the initial elementary action – in the example, the DHM is made to WALK near the object.

The information required for deciding on the applicability of an action on the one hand and the updates to the 3D scene model caused by the execution of the action on the other hand are transmitted via the interface to the 3D scene model, as described in Section 2.4 and depicted in Figure 1.

Any breakdown for (3) must ensure that the arm support, the screw driver and three screws are located near the assembly station for the car containing the center console before the tightening activity can start. (We leave it to future work to accommodate ways to carry more than two objects at the same time, using e.g. pockets.)

(4) near(arm_sppt11, station)

Different ways of reasoning are available to derive or establish goals like (4). Forward inferencing would, as long as the goal is not reached, check for which elementary actions the preconditions are fulfilled and establish the updates and postconditions, which will render further elementary actions applicable. For instance, if the DHM is not near the arm support, it may WALK to where it is located. If the DHM does not hold the arm support, it may PICK it. If the DHM is not at the station, it may CARRY the arm support to the station and PLACE it there. A widely accepted way of implementing forward reasoning involves condition-action rules parameterized by the objects and their locations, which are interpreted by a standard production system [6, 7, 8].

However, also backward inferencing is a valid approach here. Starting from a goal like (4), a chain of elementary actions is found, the final one of which has its preconditions met by the scene. We believe that an informed decision should be taken in view of a more substantial set of defined breakdowns. At the time being the breakdowns are provisionally implemented as conditional statements.

Some behavior of the CNL breakdown is configurable by parameters. For instance, one hand of the DHM can be marked as preferred. For objects to be handled by one hand, the DHM would always try to use the preferred hand. If the preferred hand is not empty, another parameter controls if the DHM would use the other hand or PLACE the object it is currently holding. This way, some personal worker preferences can be simulated.

4. Ongoing Work

While the basic mechanisms for the CNL processing and for the enrichment with information from the 3D scene and knowledge base are fully implemented and tested, two major directions are followed by ongoing work.

Just like in the case of a new production line, the definition of CNL task descriptions and related objects is being extended to cover the pilot cases. This will affect the CNL mechanisms (cf. Section 3.1) only to a small extent, as they cover a wide range of task description patterns. The semantic-pragmatic breakdown into elementary actions described in Section 3.2, however, must be defined anew for each task description.

This is a manual task that requires a deep understanding of the assembly task in order to appropriately represent the full range of its elementary actions, parameters, constraints and contextual decisions. Therefore the process of defining breakdowns will in the future be supported by specific editing tools. The breakdown definitions will have to be manipulated through a GUI, i.e. outside of the code base, which renders a clear separation of the definitions and their interpreter mandatory. The definitions will be encoded in a formal syntax with a descriptive semantics that supports manipulation by developers.

The second direction of work introduces the same requirement. User interaction during the planning workshops should be supported on at least two levels. First, the order of tasks should be changeable. This is outside of the current topic, which focuses on single task descriptions. Second, elementary actions should be added, deleted or their order be modified. These changes will be executed by non-expert users. The compatibility of pre- and postconditions of each elementary action with the respective states of the 3D scenario will be preserved. This is a necessary and sufficient condition to ensure that user modifications will not lead to ill-formed breakdowns. For instance, deleting PICK from a "PICK PICK CARRY PLACE" sequence is permitted, whereas deleting CARRY is not. Accordingly, additions and changes of order are constrained.

There will be another GUI offered to the workshop participants. They will be able to edit the result of a breakdown, i.e. a sequence of elementary actions, in the above ways. All edits will be preserved, enabling users to return to previous versions. A "permanent" edit causes the underlying rules to be modified in such a way that future breakdowns of the task description will be compatible with the latest edit state.

It remains to be investigated to which extent the GUI-induced rule modification can be carried out automatically.

5. Conclusion

In this contribution, we have suggested novel methods for connecting human language task descriptions to a level of representation designed to synthesize naturally looking, valid motion sequences that simulate assembly operations in car manufacturing. The concept of CNL is well known from language technology, and reasoning methods similar to those proposed for the breakdown are widely used in artificial intelligence systems.

Still it appears that applying the proposed interdisciplinary combination of methods to assembly planning has not been tried so far and is thus novel. It promises an unequalled level of support to both planners and workers.

The results based on four out of 22 elementary actions are encouraging. Ongoing work on the INTERACT pilot cases will tell how well the approach scales up to more complex data.

Acknowledgements

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 611007. The authors are indebted to Klaus Fischer (DFKI) and Martin Manns (Daimler AG) for numerous fruitful discussions.

References

[1] O'Brien S. Controlling controlled English. An analysis of several controlled language rule sets. The Joint Conference of the 8th International Workshop of the European Association for Machine Translation and the 4th Controlled Language Applications Workshop (EAMT/CLAW), 2003.
[2] Wojcik R., Hoard, J. Controlled languages in industry. In:. Cole R, Mariani J, Uszkoreit H, Varile G, Zaenen A, Zampolli A, editors. Survey of the state of the art in human language technology. Cambridge: Cambridge university press; 1997. p. 238-239. Available online at http://www.lt-world.org/hlt-survey/master.pdf.
[3] Funk A., Tablan V., Bontcheva K., Cunningham H., Davis B, Handschuh S. CLOnE: Controlled language for ontology editing. In: Aberer K, et al., editors. The semantic web. Proceedings of 6th international semantic web conference, 2nd Asian semantic web conference (ISWC/ASWC). Springer: Berlin, Heidelberg; 2007. p. 143-155.
[4] Kamprath C, Adolphson E, Mitamura T, Nyberg E. Controlled language for multilingual document production: Experience with Caterpillar technical English. 1998. Available online at http://www.researchgate.net
[5] Fillmore C. The case for case. In: Bach, Harms, editors. Universals in linguistic theory. New York: Holt, Rinehart, and Winston; 1968. p. 1-88.
[6] Davis R, King J. An overview of production systems. In: Elcock, Michie, editors. Machine representations of knowledge. Machine Intelligence, 8, Wiley, NY; 1977.
[7] Brownston L, Farrell R, Kant E, Martin N. Programming expert systems in OPS5. Addison-Wesley; 1985.
[8] Busemann S. Best-first surface realization. In: Scott D. editor. Proceedings of Eighth international natural language generation workshop. Herstmonceux, Univ. of Brighton; 1996. p. 101-110. Available online at http://xxx.lanl.gov/abs/cmp-lg/9605010.