# A Connectionistic Knowledge-based Approach to the Modelling and Control of Manufacturing Systems

A. Dvoryanchikova, Md. M. Hossain, A. Lobov, J. L. Martinez Lastra
Department of Production Engineering
Tampere University of Technology
Tampere, Finland
aleksandra.dvoryanchikova [at] tut.fi

I. Hammouda
Department of Software Systems
Tampere University of Technology
Tampere, Finland

***Abstract*—Traditionally manufacturing systems are controlled using Programmable Logic Controllers, which often require human intervention for system reprogramming on the arrival of a new product. In order to reduce all related costs associated with the human intervention, new systematic and automated system engineering approaches are needed. This paper introduces a knowledge-based approach to the modeling and control of manufacturing systems aiming to capture the engineering knowledge. In order to demonstrate the applicability of the methodology, a software tool was implemented and applied to the case study, a pallet-based lifter used in electronics assembly, taken from the domain of factory automation. Early experiences show that the introduced approach can be used to capture knowledge pertaining to manufacturing equipments, processes and products. The approach could be considered as a potential solution for the implementation of reconfigurable control systems.**

***Keywords - system reconfiguration, connectionistic grid, ontology, knowledge representation, factory automation, bionic approach.***

## I. INTRODUCTION

Modern manufacturing systems are developed based on Programmable Logic Controllers (PLCs), which execute control algorithms [1]. The algorithms can be described in specific industrial languages, which can be vendor specific or standardized, e.g. following the IEC61131-3 standard [1]. IEC61499 standard [2] for the modeling of distributed control systems was developed as a next step to support the development of complex industrial control systems.

The manufacturing lines controlled by PLCs are mainly built for the mass production. An introduction of a change at the factory floor in terms of new product, new equipment or new processes may require the reprogramming of the entire system. This happens because the engineering knowledge is poorly captured and/or integrated in modern engineering approaches often resulting in *ad hoc* solutions for each particular problem.

Knowledge-based approach is seen a possible solution for expressing the engineering knowledge and for facilitating the reconfigurability of automated systems. The applicability of the approach was confirmed with ontological technologies which have recently gained significant attention and widespread adoption [3]. Ontological knowledge bases were introduced to provide the semantic descriptions and to facilitate the cognition in an automated system [4]. Nevertheless, ontologies suffer from a number of drawbacks. For instance, they appear to be limited in their capability to describe processes, which are a significant and fundamental part of the technical knowledge in manufacturing. This is due to the fact that the dynamic nature of any phenomenon modeled with ontological approach is hard to capture given the rigid structure of taxonomy. In addition, expert's knowledge often has conflicting nature with many exceptions that are difficult to list in advance. Ontologies could not provide the seeking flexibility to represent expert's knowledge [4].

Ontological knowledge bases are developed in the frame of symbolic approach to Artificial Intelligence [7]. In the field of natural Cognition, there is a concurrent approach for knowledge modeling, which aroused from connectionistic theory and was introduced to explain the structure of natural knowledge [4]. In the area of Artificial Intelligence, the connectionistic approach is better known with artificial neural networks and learning algorithms [7]. In knowledge engineering, the connectionistic ideas were not fully adopted due to absence of clear formal schema for representation and then reasoning [3]. However, the connectionistic vision inspired the development of reasoning engine that is capable to reason asymmetrical structures of natural language [9], which shows the potential applicability of the connectionistic approach in knowledge representation and reasoning.

This article provides a knowledge-based perspective to acquire information of manufacturing systems regarding the modeling and control of their automated processes. Based on the connectionistic understanding of knowledge structure, the modeling principles of the systems are introduced. The model gives the semantic descriptions of the system with connectionistic concept grid (*CCG*) which is a network of the concepts. A concept is seen as a dynamically changing pattern of ongoing activations, unlike the rigid classes in taxonomies. The formal definitions are introduced with the aim to support the modeling principles and define basic elements needed for the implementation of the connectionistic approach. A software tool named Crossword has been developed to implement the connectionistic approach reported in the paper.

The rest of the paper is organized as follows. Chapter II provides the background and significance of the study. In Chapter III, the description of modeling principles and formal notations are given. Chapter IV presents the execution model. Chapter V introduces the software tool – Crossword – to allow the modeling with connectionistic grids. Chapter VI is dedicated to the case study. The paper ends with concluding remarks.

## II. Background And Significance

In general, knowledge-based approaches have been introduced to represent knowledge in a sharable and understandable way both for human and machine aiming at the replacement of engineer expertise. In the field of software systems in manufacturing, this approach has been implemented with ontologies which are claimed to give a value in three areas: unambiguous communication, industrial information infrastructure, and shared terminology and semantic alignment [9].

In manufacturing, ontologies have been widely used within the field of agent-based control in order to provide semantic description of messages exchanged by agents. Obitko & Marik [10] reviewed the use of ontologies for multi-agent systems in the manufacturing domain, and proposed the use of Description Logics (DL) as representation formalism. One of the first developments of ontologies in the field of manufacturing was the Process Specification Language (PSL), which is a First Order Logic (FOL) ontology for describing manufacturing processes developed by National Institute of Standards and Technology (NIST) [11]. Even though PSL was specifically developed for the domain of manufacturing, it has not gained widespread adoption in that area. However, it has served to influence the process models adopted by the Web Ontology Language for Services (OWL-S) ontology and especially by the Semantic Web Services Ontology (SWSO). Kulvatunyou [12] and latter Hu et al. [12] proposed the use of ontologies in conjunction with Web Services in order to integrate heterogeneous manufacturing equipment and automation software elements.

In reconfigurable manufacturing systems, the ontology components were used to create a world model and to provide the semantic descriptions [4]. The world model has included OWL-DL ontologies for processes, manufacturing equipment, products, and services. In the same work, the author presented a review of existent knowledge representation and reasoning paradigms and conclusively showed that Description Logic (DL) currently appears to be the best situated approach for providing structured descriptions of technical knowledge due to features such as knowledge reusability and computational decidability and tractability in reasoning. The most valuable inference pattern employed in DL is classification of concepts and individuals. Classification is firstly used to determine hierarchical subconcept/superconcept relationships of concepts in a terminology. Classification is also used to determine whether a given individual is an instance of a certain concept, and thus provides useful information on the properties of the individual.

However, ontologies have several drawbacks. For instance: they are restricted in process description due to their limitation in presenting the notion of time. Furthermore, ontological taxonomy is able to support only symmetric associations between classes and individuals. Also, ontologies of the same domain are challenged in registrability due to divergence of terminology. Finally, contradictory expert knowledge cannot be fully expressed with ontological formalism.

The ontological approach has its roots in the symbolic approach to knowledge structure in natural cognition. The main idea of the symbolic approach is that every object of the world is presented in the mind as a separate representation (for more details see for instance [14]). The representations are connected through the representations of higher order – "classes" [15]. All concepts are represented by nodes in a network and the links between them present associations [16]. To overcome the limitations of the symbolic approach in explanation of natural structure of knowledge, the connectionistic principles were introduced [4].

Connectionists' explanations are based on parallel distributed way of processing which was derived from the domain of neuroscience. "Parallel" here means that remembering a thing from a class, people do not search through the whole list of things from the same class as a series of short processes, but all the names of similar things are activated in parallel at the same time, therefore more common things are being remembered more quickly [1], [17].

In contrary to the symbolic representation of concepts, connectionism gives an inverse understanding of links and nodes in the network of concepts. Here the concept is seen as a chain of links which is connected to other concepts-chains through the nodes. With grow of knowledge the chain can make new links and adopt new concepts, or in opposite some concepts may leave the chain due to lost connection. Thus, the meaning of a concept depends on connections where it is currently involved rather than the place in the taxonomy.

This vision is seen beneficial in the formation of new concepts from combinations of old ones and, in addition, makes possible for the new concept to inherit the properties of "parental" concepts in a partial way. Next, connectionistic approach lets to categorize the concepts both by core and probabilistic features, which are sometimes more important. Then, a connectionistic grid can combine the several grids which are built for same domain, but are based on different criteria. [4]. The connectionistic vision may be a solution to facilitate the integration of the overlapping knowledge bases which represent the same domain and that would diminish the impact of possible errors and mismatches in semantic descriptions. The approach makes possible to relate objects in a non-symmetric way, for example to link "giver", "receiver", "given object" and "received object" [9], [4], which could help the process description. The listed features show a potential to overcome the current limitations of the knowledge-based approach and make it more flexible.

To control and reflect the events, states and processes of the manufacturing system, a model based on the connectionistic vision was introduced. The model is a network of interpenetrating concepts. In contrast to ontologies, concept is understood as a dynamically changing pattern of the grid, which discovers itself through ongoing activation, rather than a

rigid class of objects. The activation in the model registers the events and leads to the actions in the real world. In the next section, modeling principles of a connectionistic framework are discussed.

## III. CONNECTIONISTIC FRAMEWORK

### 3.1 Set of knowledge

The whole manufacturing process can be described in terms of three domains of knowledge: Equipment, Process and Product [4]. The concepts of the domains Equipment and Product are represented by nouns of natural language and the concepts of the domain Process are represented by verbs. Intersecting each other, the concepts of all three domains form the concept grid for an automated system.

*CCG* supports the event-based approach, thus representing and controlling the current events and states of the system. The interesting concepts of different branch of knowledge are ordered in simple structures like SUBJECT – PREDICATE – OBJECT. Figure 1 shows a simple grid that could represent a trivial operation of gripping a tool.

To express an event or a state of the system, the grid has to include minimum three concepts of at least two domains. One of the domain has to be the Process and another one or two have to belong to the Equipment or/and the Product (Fig. 1). No grid is possible without a concept of Process; and no grid is possible with only concepts of Process. A simple grid can be built out of three concepts and two intersections between them: one of the concepts is from the domain of Process and the two others are from the domains of Equipment and/or Product.

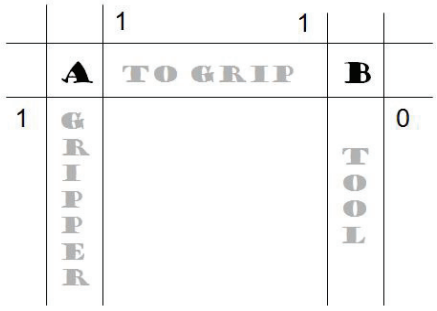


Figure 1. A grid of two domains (Equipment and Process) with the bidirectional indexes of intersections.

It is important to express and reason on the asymmetrical relations between objects. In Fig. 1 three concepts – Gripper, To Grip, and Tool – are involved in the interaction. A situation where a gripper could grip a tool is possible in the real world, but the opposite case is impossible. In order to express asymmetrical relations, the bidirectional indexes of intersections are introduced. In every moment of time every intersection has a 2-digit number which shows the direction of the relation. The concept acting as a "subject" of the statement is indicated with "1". A verb in active form is indicated with "1". The "object" of the action is indicated with "0". A passive verb is indicated with"0". Thus every involved concept brings one digit to the bidirectional indicator of an intersection. For instance, to present and reason on the operation "to grip the tool with the gripper" (Fig. 1), the bidirectional index of intersection A between 'Gripper' and 'To Grip' is "11" since both concepts are active. The index of intersection B between 'To Grip' and 'Tool' is "10" as 'Tool' is an object acted upon.

The focus of this paper is on assembly processes, which is a part of manufacturing. In this domain the functionality of the system can be expressed with three levels of abstraction: Assembly Task, Assembly Process, and Assembly Operation [19]. In *CCG*, starting from Assembly Task up to Assembly Operation, every domain of manufacturing knowledge (Equipment, Process and Product) is represented with greater number of details. For instance, the concept 'Robot' has structural division to Joint and End-effector (Fig. 2). Thus a concept of every domain of knowledge can be disclosed with more details with the semantics of the particular domain in order to provide the functionality in all three levels of abstraction of the assembly processes.

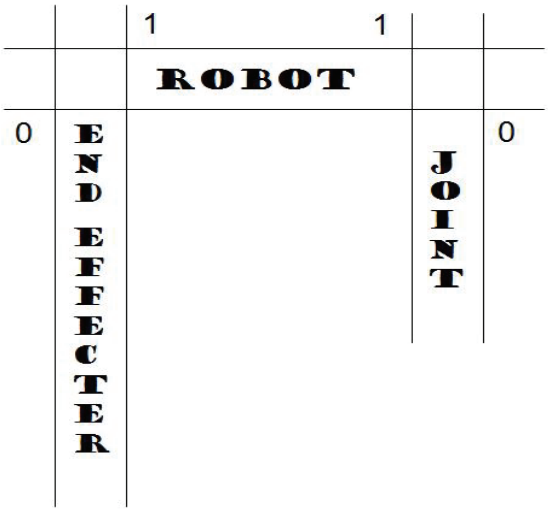


Figure 2. Representation of the domain in levels of manufacturing process

Following the connectionistic vision, the meaning of a concept is a pattern of activation which is highlighted by the parts of grid. The pattern is dynamically changing and in each particular moment of time the number of instances involved to the activation can differ. In the extreme cases, a concept can be represented with only one strip of the conceptual grid, or in the opposite a concept can be built of the entire conceptual grid (for instance, a concept 'System'). In Fig. 2 concept 'Robot' is currently represented with three strips {Robot, Joint, End-effector}. The strip 'Robot' is a core of concept (*CoC*). Graphically *CoC* is presented as a single strip in the grid, while the concept is a complex pattern of activation. The configuration of the concept may change with time, but the pattern of the configuration will always include *CoC*. The dynamically changing pattern reflects and initiates changes in the system.

The concepts of same domain but with different levels of abstraction are ordered with the 2-digit indexes. For instance, in Fig. 2, both intersections have indexes "10", where "1" is at the level of *CoC* 'Robot' (level of Assembly Task) and "0" is at the level of 'Robot's details (level of Assembly Processes).

3.2 Formal notations

This subsection provides the basic definitions for the connectionistic approach, which defines necessary structure to make it possible to implement the methodology.

The connectionistic approach deals with the concepts represented as a set of activated connections of the grid. The connectionistic concept grid *CCG* can be formally represented as a tuple:

$$CCG = \langle CoC, A, P, AA, \alpha \rangle \qquad (1.)$$

where:

- $CoC = \{c_1, c_2, \ldots, c_n\}$ is a finite set of cores of the concepts
- $A = \{a_1, a_2, \ldots, a_m\}$ is a finite set of activations defined for the *CCG* (representing actual concepts in the sense of symbolic approaches)
- $P \subseteq 2^{CoC}$ is a set of the activation patterns that are defined for the given *CCG*
- $AA \subseteq A$ is a set of currently enabled activations of the *CCG*
- $\alpha: A \rightarrow P$, is an activation function that maps an element of set *A* to the element of set *P*.

Set *A* can be dynamic in order to allow the learning process of the grid. For this the learning function must be specified for the grid. The learning concept grid *LCCG* can be defined as follows:

$$LCCG = \langle CCG, \lambda \rangle \qquad (2.)$$

where:

- *CCG* is a concept grid as defined in (1.)
- $\lambda: 2^{AA} \rightarrow a_{New}, a_{New} \notin A, A = A \cup \{a_{New}\}$; is a mapping of some active activations to the new activation that is added to the former activations set A to form a new activations set.

In the realization of the connectionistic grid tool, the elements of the enabled activations set *AA* are mapped to the actions expressed in terms of I/Os of the controlled process and/or in terms of other activations.

The activations evolution law can be described as a function of currently enabled activations:

$$AA' = activate(AA) \qquad (3.)$$

The activate function takes as an input a set of currently enabled activations *AA* and derives a new set *AA'*.

For the control of the physical process, the set of process inputs *I* and outputs *O* have to be specified. The definition of the *CCG* (1.) to address the control of the process can be extended. Control Connectionistic Concept Grid *CtrlG* is defined as a tuple:

$$CtrlG = \langle CCG, I, O, \iota, \omega \rangle \qquad (4.)$$

where:

- *CCG* is a connectionistic concept grid as defined in (1.)
- $I = \{i_1, i_2, \ldots, i_k\}$ is a set of control inputs
- $O = \{o_1, o_2, o_s\}$ is a set of control outputs
- $\iota: I \rightarrow AA$ is an input function that maps the control inputs to the activations.
- $\omega: AA \rightarrow O$ is the output function that maps the enabled activations of *CCG* to the control outputs.

The enabled activations of *CtrlG* are evolving through *activate* function (3.) and input/output functions defined in (4.)

## IV. Execution Model

Considering the equations (3.) and (4.), *activate* and $\iota$ and $\omega$ functions are implemented in execution engine (Fig. 4).

The factory automation system (FAS), which consists of different hardware and software components from different manufacturers using many standards and protocols, requires a logical hierarchy. In such layered architecture, each layer interacts with the other through certain interface using certain protocol. A chain of executions takes place along the different layers. Fig. 4 shows the execution chain model of three layers: Model, Execution engine and Physical process.

In the top layer, a "Model" is given in terms of a connectionistic grid. It is assumed that the grid is input via the visual interface of a software tool. The middle layer, which is also a software layer, acts as an interface between the high level model and the underlying physical equipments. This layer has the instruction set; using these instructions it can instruct the machines to execute the activation of the high level model, $\iota$, and it can also report the status of the physical equipments to the model by changing the activation of the model, $\omega$.

The bottom layer is the physical equipment layer, where all the machines of the manufacturing system lye. This layer takes command from the execution engine, causes the equipments to work accordingly, and reports the status of the physical equipments through well-defined I/O operations.

Fig. 3 shows six steps involved in the execution chain of a factory automation system:

1. Activate the network (instruction given by the user).
2. The activation of certain part of the model is reported to the execution engine.
3. Execution engine converts the high level command to execute the activation for the physical equipment.

4. After executing the command, the physical equipment reports the status through I/O operations to the execution engine.

5. Execution engine reports the status of the physical equipments by changing the activations of different concepts in the model.

6 a. The user can observe the current status of the system and the system response in terms of activation of different concepts in the model.

6 b. The feedback from the hardware through Execution engine may activate new concept intersection, which will initiate the execution of the chain again.

In the next section, tool support for *CCG* and the execution chain shown in Fig. 3 is presented.

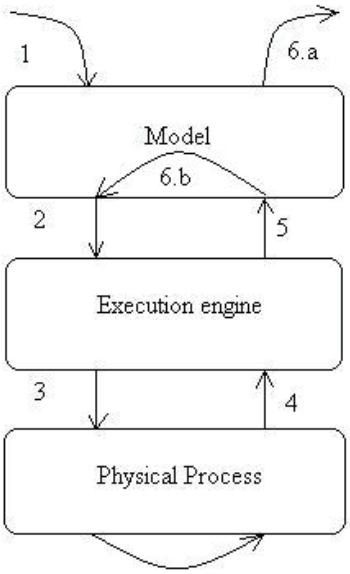


Figure 3. Model execution chain of FAS

## V. TOOL SUPPORT: CROSSWORD

In order to demonstrate the connectionistic approach described in this work, a software tool named Crossroad has been developed. The tool, which is developed as a desktop application, is intended to interact with the PLCs of the manufacturing equipments by implementing the steps in the execution chain depicted in Fig. 4. The architecture of the tool is based on the Model View Controller (MVC) architectural style. The view part is responsible for the visual interface display. The model contains the CCG data. The controller communicates with view, calculates model data and interfaces with the "Execution Engine".

Fig. 4 shows the modularization of Crossroad: GUI module for the user interface, communication module for tool to machine's PLC communication, inference module for concept and intersection reasoning from the defined rules and the controller module for controlling all the module and work in a synchronized manner. In addition, the figure shows the packaging of the code base.

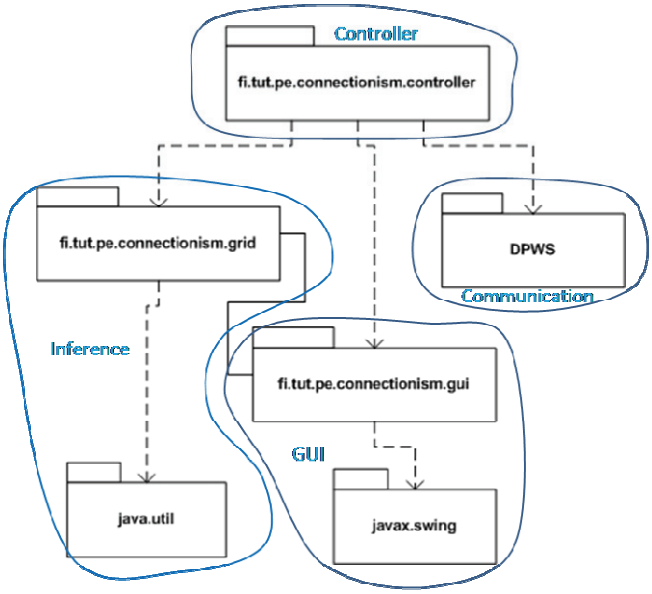


Figure 4. Package diagram of the tool

Package *grid* holds the model and package *gui* holds the view classes. There is a link between these two packages because some visual representation value is transferred into the model. The *controller* package uses all three packages. The *DPWS* (Device Profile for Web Services) package is a third party open source java implementation [20] for the communication with the physical process. The tool has been implemented in Java and the swing library has been used for drawing the concepts and intersections in the visual interface.

Using Crossword, the user constructs a model in terms of a connectionistic grid. In addition, the user provides validation rules for the allowed combinations of activations. The tool is then capable to validate the input model against the rules. In case of a violation, the user is notified via an informal message. For example, if a validation rule states that a pallet cannot be lifted up in the middle of loading a product on it; then if the user defines a model where the motor for lifting the pallet can be started in the middle of the pallet uploading, the violation is caught and reported to the user. The tool implements equation 1 shown earlier, by representing the set of core of concepts CoC and the set of activations $A$. In addition, the tool calculates the set of concepts and intersections evolving dynamically.

In the next section, the interface of Crossroad is discussed in the context of a case study.

## VI. CASE STUDY

An industrial lifter example shown in Fig. 5 is used for the case study. The lifter is designed to work in a two-level conveyor system. Pallets may circulate in such a system transferred by so-called "Start" and "End" lifters between the levels. In this lifter there is a sledge conveyor, which can move the pallet to any direction and two other conveyor segments which can either upload the product or download it, thus causing it to move in only one direction. The lifter's responsibility is to transfer the pallets between the conveyer

levels. Thus it provides the pallet circulation between the conveyor levels.

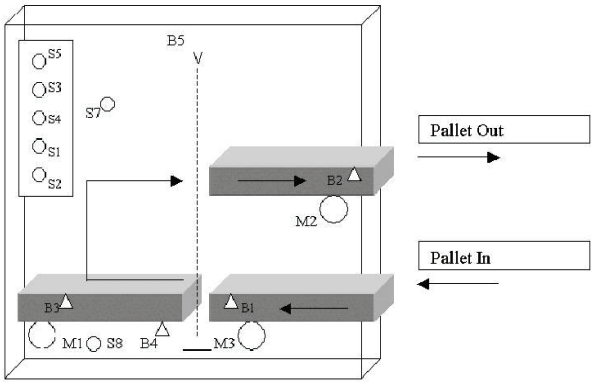


Figure 5. Functional components of the lifter [21]

Fig. 6 depicts a connectionistic grid for the lifter example. For demonstration purposes, only a small part of the model is shown. To understand the whole scenario, consider that a pallet has been downloaded to the lower conveyer segment of the lifter and the pallet is to be lifted up to the upper lever conveyer. The Crossword tool has been used to build the lifter connectionistic grid. The Crossroad implementation of the example grid is shown in Fig. 8.

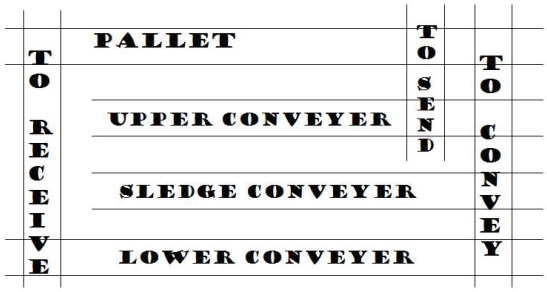


Figure 6. *CCG* model showing the lifter components

The user has to create a project for every *CCG*. Then the concepts and intersections are drawn according to the need of the underlying hardware. After that the intersections are activated and the execution of the grid model starts. In Fig. 7, the CoC Lower_ conveyor has been selected in the grid. The left view shows a tree layout for the available concepts and intersections. The bottom view lists the properties for each concept. For instance, the view shows that the Lower_conveyor intersects with To_receive and To_convey.

Let us denote the "lower terminal" as c1, "upper terminal" as c2, "sledge" – c3, "to send" – c4, "to receive" – c5, "to convey" – c6, "pallet" – c7. Therefore a *CoC* set (1) takes the following form {c1, c2, c3, c4, c5, c6, c7}. The lifter should be capable of performing the following processes, which are encoded as activations: lifter loading process (a1), pallet transportation from the lower terminal to the sledge (a2), vertical motion of the sledge with the pallet (a3), vertical motion of the sledge without a pallet (a4), pallet unload from the sledge to the upper terminal (a5), pallet unload from the upper terminal to the next piece of equipment connected to the lifter (a6). Therefore the set of activations *A* looks as follows {a1, a2, a3, a4, a5, a6}. The activations are defined in terms of elements of the power set of *CoC*. The activations are defined as follows: a1 = {c1, c5, c7}; a2 = {c1, c3, c7, c6}; a3 = {c3, c7}; a4 = {c3}; a5 = {c2, c3, c6, c7}; a6 = {c2, c4, c7}.

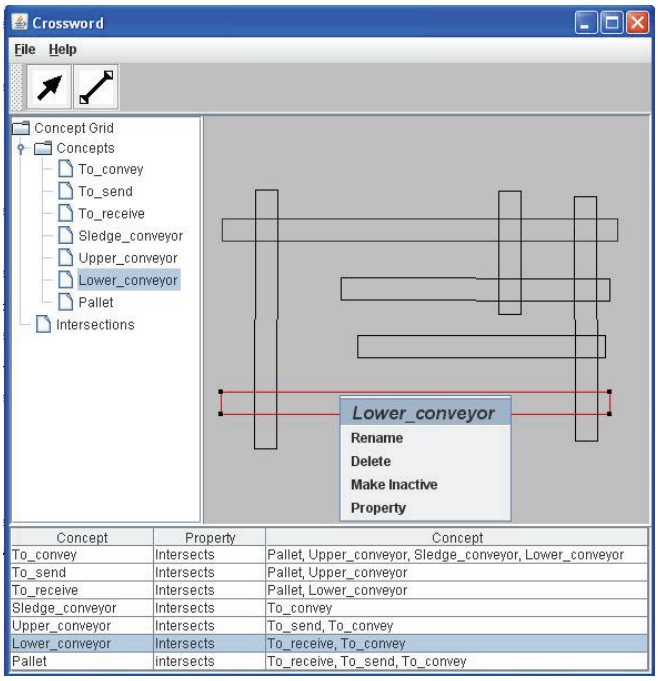


Figure 7. *CCG* model shown in the tool's interface

The change between the activations is performed through *activate* (3), and $\iota$ and $\omega$ functions (4). The pallet being loaded to the lower segment is denoted as a1. The pallet being loaded to the lower segment is denoted as a1. In order to continue the transportation process of the pallet, the *activate* function, which is responsible to derive new activations, returns the next activation as the state of the grid is updated by $\omega$. Thus, once the pallet is loaded, *activate* function should return the next combination of the elements in *CoC*, which is a2.

## VII. CONCLUSION

A knowledge-based approach with inspiration driven from the connectionistic theory was proposed for the modeling and controlling of automated systems. The approach provides a formal but still flexible structure of knowledge – *CCG*, which may help to overcome current limitations in expression of experts' knowledge. A set of modeling principles and formal notations was developed for the implementation of the approach. A prototype tool named Crossword has been developed to demonstrate the applicability of the knowledge-based approach. The software was applied to the case study of a pallet-based lifter used in electronic assembly. The initial results demonstrate that the introduced approach can be used to capture engineering knowledge pertaining to manufacturing process. The approach could be considered as a potential solution for the implementation of reconfigurable control systems.

In order to adapt the approach to the field of manufacturing control, a number of research directions must be explored. First the principles for knowledge reasoning in *CCG* have to be further clarified. At the implementation level, the Crossword tool should be further developed and tested with more elaborated cases including a testbed of a production line for electronics assembly. In addition, the usability of the tool has to be explored more intensively.

## REFERENCES


[1] H. Jack, Automated Manufacturing Systems with PLCs, Version 5.2, September 2008

[2] R. W. Lewis, Programming industrial control systems using IEC 1131-3. The Institution of Electrical Engineers, ISBN 0-75296-950-3, London, 1998.

[3] International Electrotechnical Commission, IEC TC65/WG6, "Committee Draft: Function Blocks for Industrial-Process Measurement and Control Systems - Part 1 Architecture", International Electrotechnical Commission, 2003.

[4] A. Gomez-Perez, M. Fernandez-Lopez, O. Corcho, Ontological Engineering: With Examples from the Areas of Knowledge management, E-Commerce and Semantic Web. Springer-Verlag, London, 2004.

[5] I. M. Delamer, Event Based middleware for Reconfigurable Manufacturing Systems: A Semantic Web Services Approach. Doctoral Thesis, Tampere University of Technology, Finland, 2006.

[6] D. Deneux, and X. H. Wang, "A knowledge model for functional re-design", Engineering Applications of Artificial Intelligence, 2000, 13, pp. 85-98.

[7] B. Lorenz, and E. Barnard, "A brief overview of artificial intelligence focusing on computational models of emotions", In Conference Proceedings 1st International Engineering & Neuro-Psychoanalysis Forum, ENF 2007, Vienna, Austria, July 23 2007, pp. 1-12.

[8] N. Goldblum, The Brain-Shaped Mind. Cambridge University Press, 2001.

[9] L. Shastri. "Advances in SHRUTI - A Neurally Motivated Model of Relational Knowledge Representation and Rapid Inference Using Temporal Synchrony", Applied Intelligence, 1999, 11 (1), pp. 79-108.

[10] C. Schlenoff, R. Ivester, D. Libes, P. Denno, and S. Szykman, "An analysis of existing ontological systems for applications in manufacturing and healthcare", NISTIR 6301, National Institute of Standards and Technology, Gaithersburg, MD, 1999. W. Bechtel, and A. Abrahamsen. Connectionism and the mind: an introduction to parallel processing in networks. Cambridge, MA: Basil Blackwell, 1991.

[11] M. Obitko, and V. Marik, "Ontologies for multi-agent systems in manufacturing domain", Proceedings of the 13th international workshop on database and expert systems applications – DEXA'02, Aix-en-Provence, France, 2002 September 2-6, pp. 597-602.

[12] C. Schlenoff, M. Gruninger, F. Tissot, J. Valois, J. Lubell, and J. Lee, "The Process Specification Language (PSL): Overview and Version 1.0 Specification", NISTIR 6459. National Institute of Standards and Technology, Gaithersburg, MD, 2000.

[13] B. Kulvatunyou, and N. Ivezic, "Semantic Web for Manufacturing Web Services", In processing of 5th Biannual World Automation Congress, 2002 June 9-13, Orlando, FL, USA, 14, pp. 597-606.

[14] Z. Hu, E. Kruse, and L. Draws, "Intelligent Binding in the Engineering of Automation Systems using Ontology and Web Services", IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews, 2003, 33(3), pp. 403-412.

[15] E. L. Rissland, "Artificial Intelligence: knowledge representation", Ch. 4 in Cognitive science: an introduction, second printing, Stillings, N. (Ed.). Massachusetts Institute of Technology, 1987.

[16] E. Rosch, and C. B. Mervis, "Family resemblances: studies in the internal structure of categories", Cognitive Psychology, 1975, 7, pp. 573-605.

[17] A. Collins, and E.F. Loftus, "A spreading activation theory of semantic processing", Psychological Review. 1075, 82, pp. 407-428.

[18] D. Rumelhart, and J. McClelland, Parallel distributed processing: explorations in the microstructure of cognition. Cambridge, MA: MIT Press, 1986.

[19] W. Bechtel, and A. Abrahamsen, Connectionism and the mind: an introdiuction to parallel processing in networks. Cambridge, MA: Basil Blackwell, 1991.

[20] J. L. Martinez Lastra, Reference mechatronic architecture for actor-based assembly systems. TTY-Paino, Tampere, 2004.

[21] SOA4D, DPWS implementation, https://forge.soa4d.org/, [Accessed: November 2008].

[22] A. Lobov, "An approach the formal verification of automated manufacturing systems with programmable control", M.Sc. Thesis, Tampere University of Technology, 2004.