A k-Nearest Neighbour Technique for Experience-Based Adaptation of Assembly Stations

Daniele Scrimieri · **Svetan M. Ratchev**

Received: date / Accepted: date

Abstract We present a technique for automatically acquiring operational knowledge on how to adapt assembly systems to new production demands or recover from disruptions. Dealing with changes and disruptions affecting an assembly station is a complex process which requires deep knowledge of the assembly process, the product being assembled and the adopted technologies. Shop floor operators typically perform a series of adjustments by trial and error until the expected results in terms of performance and quality are achieved. With the proposed approach such adjustments are captured and their effect on the station is measured. Adaptation knowledge is then derived by generalising from individual cases using a variant of the k-nearest neighbour algorithm. The operator is informed about potential adaptations whenever the station enters a state similar to one contained in the experience base, that is, a state on which adaptation information has been captured. A case study is presented, showing how the technique enables to reduce adaptation times. The general system architecture in which the technique has been implemented is described, including the role of the different software components and their interactions.

Keywords Knowledge-based systems · k-nearest neighbour algorithm · Assembly · Architectures

1 Introduction

Dealing with disruptions, reconfiguring or simply enhancing the performance of an assembly station requires knowledge of the process which is difficult to define formally. Engineers and shop-floor operators make decisions on changes and adaptations based on their experience. Typically they acquire experience by experimenting with the machine, carrying out a trial and error process which is difficult to predict and plan. Knowledge built this way is not formalised and cannot be easily transferred among operators and to new production environments.

A method for capturing knowledge about adaptations is required to avoid human knowledge loss and enable sharing and reuse. Reuse of adaptation knowledge facilitates problem analysis and solving, supports decision making and guarantees higher responsiveness to disruptions and changes. Sharing operators' experience on machines and processes allows to train operators to work in different production scenarios. Specifically, an operator performing an adaptation can be provided with details about what other operators did in similar contexts and the outcome of their actions.

This paper presents an approach to automatically capturing information on adaptations being performed and deriving knowledge. Adjustments to the system are recorded and related to the current system state. The performance of the system is measured before and after the change in order to determine whether it has had a positive impact. This form of experience on adaptations is constantly accumulated and the feedback provided by the performance measurements allows to learn how to adapt effectively the system.

When there is a disruption that has already occurred in the past, an adjustment that has proven to be successful for that specific disruption or a similar one is recommended to the operator. Not only are successful adjustments recorded, but also adjustments that failed. This is useful in order to avoid reapplying those changes that did not cause any improvement.

Recommended adjustments are not applied automatically. Instead, a ranked list of adjustments to choose from is presented to the operator, generated based on the available experience on the problem. The operator will then select one,

D. Scrimieri · S. M. Ratchev
Department of Mechanical, Materials and Manufacturing Engineering
University of Nottingham, Nottingham, NG7 2RD, UK
E-mail: {Daniele.Scrimieri,Svetan.Ratchev}@nottingham.ac.uk

normally the one ranked first, apply it and evaluate the resulting state. If the effect is not satisfactory, a different adjustment from the list can be tried out.

This paper is an extended version of Scrimieri and Ratchev (2013), presented at the 11th IFAC Workshop on Intelligent Manufacturing Systems, São Paulo, Brazil, 22-24 May 2013. The new contribution of this paper is the presentation of a case study and experimental results.

The paper is organised as follows. Section 2 contains some background information on learning and adaptation. Section 3 describes how experience on adaptations is captured. Section 4 presents a technique for providing decision support on adaptations, based on experience. Section 5 contains details on the software architecture. Section 6 presents a case study and experimental results. Sections 7 and 8 contain, respectively, a discussion with ideas for future work and conclusions.

2 Background

Adaptation knowledge evolves and is updated by continuous learning. This section summarises some approaches to learning and adaptation in production systems, including assembly systems.

2.1 Learning

Knowledge acquired in a manufacturing organisation can be measured with learning curves (Jaber, 2011). Learning curves can estimate costs based on the cumulative volume of production. The assumption is that costs decrease as cumulative production increases. This is justified by the fact that the more an organisation produces the more it learns how to produce and becomes efficient.

If a product to manufacture is similar to another product produced in the past, resources employed and knowledge built for the production of the old product can be transferred to the new one. The old process may be adapted. Therefore, the starting costs will be lower than those that would be incurred if there was no prior knowledge.

Experimentation is another form of learning. Changes to the process are made and the results are evaluated and compared to the expectations. This is radically different from the learning-by-doing underling learning curve theories, where learning seems to happen autonomously.

Foguem et al (2008) formalise experience feedback in the context of continuous improvement of industrial systems. The experience feedback process is modelled with conceptual graphs and an ontology is built to define experiential elements. The aim is to transform experience feedback into explicit knowledge and know-how that can be shared among the involved actors.

A particular stage of the life-cycle of production systems in which learning occurs is the ramp-up phase. This phase represents the period of time it takes for a new production system to reach high levels of production and product quality (Korena et al, 1999). It has the following characteristics (Surbier, 2010):

- The production capacity is low;
- There are frequent disturbances regarding the equipment, the quality of the product and the supply chain;
- The initial level of knowledge about the product and the process is low.

When the ramp-up phase starts the production process is not entirely understood and discrepancies hinder efficiency. The initial process recipe is not yet suitable for large volumes of production. Although changes are required, they may conflict with learning. Terwiesch and Xu (2004) consider learning as the process of eliminating the discrepancies between process specification and execution during ramp-up. They analyse the trade-off between applying process changes immediately and accomplishing learning tasks. A ramp-up strategy called "copy-exactly" is proposed, which freezes the process recipe for a certain period of time (i.e. does not allow changes during it) to enable learning.

Fjällström et al (2009) analyse the types of information needed to deal with critical events during ramp-up. The authors consider problem, domain and problem solving information in a case study in the automotive industry. They find out that problem and domain information are the most important. In addition, the domain information required to respond to a critical event is not exclusively related to the category of the event. For example, handling a critical event about a product may require domain information not only about the product itself, but also about the process and the equipment.

2.2 Adaptation

Flexible manufacturing systems can adapt themselves to produce anticipated product variations. This level of adaptability is achieved a priori during design as opposed to when new production demands arise. In contrast, reconfigurable manufacturing systems can be adapted to new volumes of production and new products. Their modular and adjustable structure enables to change the hardware or control software and offer new capacities and functionalities (ElMaraghy, 2006; Korena et al, 1999).

Two types of adaptations can be identified: static adaptations performed "off-line" when the system is not running, consisting of long-term physical changes (e.g. installation of new equipment), and dynamic adaptations performed "on-line" when the system is running, consisting of short-term changes to alter, for example, scheduling or routing.

Static adaptations are normally planned by human experts or through automatic planning methods, while dynamic adaptations are carried out autonomously by self-organising systems able to react to changing environments (Järvenpää, 2012; Westkämper, 2006). A capability-based adaptation methodology that matches product requirements and available system capabilities is proposed by Järvenpää (2012) to support both human planning and reactive systems.

The multi-agent and holonic approaches are often used for dynamic adaptation, to make logical or parametric changes online. Self-managing evolvable assembly systems (Frei et al, 2009) can self-adapt at production time by changing parameters of their constituent modules, the distribution of tasks among modules or the physical layout (e.g. adding, removing or moving modules). The system is controlled by a set of distributed agents, each with certain capabilities. Agents can form coalitions which can change dynamically to adapt to new conditions. The automatic identification and configuration of new devices is inspired by the plug-and-produce holonic system of Arai et al (2001).

Oates et al (2012) and Scrimieri et al (2013) describe some automated techniques specifically designed for supporting the operator during the ramp-up of an assembly system. Two k-nearest neighbour algorithms are used to classify machine states, indicating which adjustment to perform and which value to set. These works address in particular the following challenges:

- The dimensionality of the data constantly changes. In fact, the number of parameters used to describe machine states changes as modules or sensors are added or removed;
- The training set may not be large enough to generalise from;
- The experience should be transferable among similar machines.

3 Adaptation Contexts and Experience

By adaptation context we mean the state of the assembly station when an adaptation is performed. A state is defined by the process recipe in use, which specifies all process parameters, and the process status variables obtained by processing sensor data.

We assume that a KPI (key performance indicator) can be calculated on process variables. A KPI represents a manufacturer-specific measure of performance and allows to detect disruptions and evaluate the effect of an adaptation, thus determining whether an adaptation has had a positive effect. Typically a KPI is a function of volume of production and quality, and is defined during system specification or design. At the same time, a target value is set. This value may be chosen based on the KPI of a similar machine. During the ramp-up phase, the production process is constantly improved until the target value is reached.

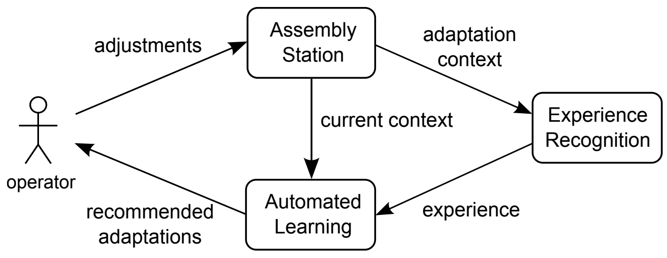

Fig. 1 Flow of information between the components and actors involved

In our software implementation, process variables are encapsulated in events that are transmitted across a distributed architecture. There are two types of events: status events and adjustment events. Status events contain, for example, cycle times or quality measurements of the parts being assembled. They depend on the modules and sensors installed. If a module or sensor is removed, the information that it supplies will no longer be available. This increases the complexity of the similarity measure between states (Oates et al, 2012). Status events are stored in an event base that can be queried for the events generated in a specific time period. The event base allows to construct a representation of the machine state at a specific point in the past and is used in the experience creation process.

Adjustment events are generated when the operator adjusts the system. Adjustments can be performed either via software or manually, that is, by making physical changes to the machine. In the latter case, details of the change must be entered into the system through a human-machine interface. Examples of adjustments include changing pick-and-place points, the pressure of grippers, the sequence of operations, the pallet geometry, etc.

An adjustment is typically an atomic change that cannot be split into several individual changes. Compound adjustments can also be made, although in this case it is not possible to assess the effect of each single adjustment separately from the others. Adjustments can be specified at different level of detail. For example, an adjustment about changing the pressure of a gripper could be specified as "change the pressure", "increase the pressure", "set the pressure to x", and so on. We assume that a finite set of adjustments is defined, where each adjustment is not a generalisation or specialisation of any others.

Fig. 1 illustrates the flow of information between the components and actors involved in the adaptation process. The Experience Recognition component is presented in Section 3.1, while the Automated Learning component is described in Section 4.

3.1 Experience Recognition

Adjustment events trigger the creation of new experience. Experience represents changes to the system, including the context in which they are made and the impact that they have. Experience can be related to either successful or unsuccessful adaptations, where the outcome is determined based on the KPI. If the KPI of the machine state after the change is higher than that of the initial machine state (i.e. the performance has increased), the adaptation can be considered successful.

The Experience Recognition component listens for adjustment events and creates new experience when one is received. The experience creation process consists of the following operations:

1. Query the event base for the last status events generated before the adjustment;
2. Construct a representation of the machine state $\mathbf{x}_1$ before the adjustment, using the status information obtained in the previous step and the process parameters in use before the adjustment;
3. Calculate the KPI on $\mathbf{x}_1$;
4. Wait for the effect of the adjustment to be measurable on the system;
5. Query the event base for the latest status events;
6. Construct a representation of the machine state $\mathbf{x}_2$ after the adjustment, using the status information obtained in the previous step and the new process parameters;
7. Calculate the KPI on $\mathbf{x}_2$.

In particular, the duration of step 4 depends on the specific adjustment performed. Experience is stored in an experience base that is accessed when decision support is provided to the operator. An experience instance can be represented in the form

$$(\mathbf{x}_1, kpi(\mathbf{x}_1), adj, \mathbf{x}_2, kpi(\mathbf{x}_2))$$

where *kpi* is the KPI function and *adj* is the triggering adjustment. The machine state $\mathbf{x}_1$ is the adaptation context of the experience instance.

4 Generation and Application of Adaptation Knowledge

When the operator requests support in a given adaptation context, adaptation knowledge is derived from the experience. If the experience base contains instances which have an initial state similar to the current adaptation context, these instances can be used to aid the operator. In particular, the experience base is searched for the most similar context and the adjustment with the best result.

The similarity is measured with a distance function calculated on contexts. Similar contexts are likely to capture similar disruptions or adaptations. If we have some knowledge on the changes made in one of them, then we can use this knowledge in the other one to predict the impact of those changes on the performance. This approach is general and does not depend on the particular disruption or adaptation. However, the accuracy of such a prediction is heavily dependent on the representation of states, the KPI and the similarity measure.

The adaptation context in which decision support is required is constructed similarly to experience. The event base is queried for the latest status events. The current process recipe in use and the values of the process variables obtained from status events represent the current adaptation context.

4.1 Adaptation as Classification

The adaptations to recommend are determined by applying a classification technique to the current adaptation context. Classification is a machine learning problem whose aim is to infer the class of a given instance. Instances are described by a set of attribute-value pairs and can be partitioned into classes. The class of an instance is identified based on a given set of examples, called *training set*.

In our problem, a class is represented by all the adaptation contexts that can benefit from the same adjustment, and it will be identified by the adjustment itself. The examples are given by the adaptations contained in the experience base and the instances to classify are the adaptation contexts where decision support is requested.

Some adjustments may not be applicable in a certain context. This is the case, for example, when an adjustment has already been applied, it is related to a device that is not currently connected to the station, or some other preconditions do not hold. Thus, the set of potential classes are dependent on the specific adaptation context.

Every example in the training set consists of a input vector and a discrete output value. The output value represents the class of the example. A classification algorithm normally analyses the training set, generalises the association between input vectors and output values, and constructs a *classifier*, that is, a program that predicts the correct class of instances not previously seen (i.e. not in the training set).

The technique that we are going to describe generalises not only based on the association between input vector and output, but also by estimating the impact of an adaptation. Estimates are determined by calculating the difference in KPI in the corresponding experience instances.

Classification is a type of supervised learning, which is about learning from labelled examples. (The other type is regression, where the output value of the examples is continuous and the output of the learning task is a regression function). Common classification techniques include support vector machines, decision trees, neural networks and k-nearest neighbour. Applications include classification of faults in power distribution networks (Lazzaretti et al, 2013), classification of power quality disturbances (Barbosa and Ferreira, 2013) and transmission line protection (de Carvalho et al, 2014).

Machine learning techniques have the advantage that they derive knowledge directly from data. There is no need for eliciting knowledge from experts and formalise it in declarative or operational language. In addition, they are effective also with unforeseen cases.

4.2 Search in the Instance Space

In the k-nearest neighbour algorithm (kNN), the instance to classify is assigned a class which is a function of the classes of the k nearest neighbours. Both the examples and the instances to classify are vectors of a multidimensional feature space. Distances between vectors are calculated, for example, using the Euclidean distance, if the values are continuous. If $k = 1$ the instance is assigned the class of the nearest neighbour in the training set. If $k > 1$ a voting scheme must be adopted to select a class among those of the k nearest neighbours. The simplest scheme is the one that selects the most common class.

In contrast with other classification techniques, the kNN algorithm does not explicitly construct a classifier in the form of a decision tree or some other structure. All the examples are kept in memory, in our case in the experience base. The inductive generalisation does not occur when instances are added to the training set. Instead, it takes place when an instance has to be classified. This is why kNN techniques are referred to as examples of lazy learning.

There is no computational overhead when new experience is generated, as it is simply added to the experience base and no classification structure needs updating. This is particularly suitable for an assembly system that undergoes numerous changes, thereby generating new experience, for example during reconfiguration and ramp-up.

Several variants of the kNN algorithm have been developed with different distance functions, voting schemes or dynamically weighting neighbours. Application domains include shape recognition (Belongie et al, 2001), protein structure prediction (Kim, 2003) and database retrieval (Basri et al, 2011).

4.3 Similarity Between Machine States

Let $\mathbf{x}$ and $\mathbf{y}$ be two machine states over n variables (remember that the number of variables in two states can be different as it is dependent on the modules and sensors installed at a specific time). The distance between $\mathbf{x}$ and $\mathbf{y}$, denoted by $d(\mathbf{x},\mathbf{y})$, is defined as:

$$d(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{i=0}^{n} d_i(x_i,y_i)} \tag{1}$$

where $d_i(x_i,y_i)$ is defined as $\frac{(x_i - y_i)^2}{range_i}$ ($range_i$ is the range of variable i) if variable i is numerical. Otherwise (attribute i is categorical), $d_i(x_i,y_i)$ is defined as 1 if $x_i \neq y_i$, else 0. If an attribute is missing in one or both states, then $d_i(x_i,y_i)$ yields 1. This is the *heterogeneous euclidean-overlap metric* (Wilson and Martinez, 1997) and we use it to measure the similarity between machine states.

Attributes may not all have the same influence in determining the similarity between two states. Some attributes can be noisy, irrelevant or redundant. To cope with these problems, different weights can be assigned to attributes. Equation 1 then becomes:

$$d(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{i=0}^{n} w_i d_i(x_i,y_i)} \tag{2}$$

Wilson and Martinez (1996) present an evolutionary algorithm for determining a set of weights that allows to obtain estimates of accuracy significantly higher than those of non-weighted distances.

4.4 Estimated Effect of Adaptations

Searching the experience base for neighbours allows to find which adaptations were performed in the past in similar contexts. However, experience contains not only successful adaptations, but also adaptations that failed. Also, the effect of an adaptation can be measured by calculating the KPI on the relative experience. Thus, while searching the experience, in addition to the similarity between contexts we take into account the effect of adaptations, as measured by the KPI.

Suppose that an adaptation context $\mathbf{x}$ is to be classified based on the stored experience. An estimate of the impact on performance caused by an adjustment *adj* in the experience is given by the weighted mean of the difference in KPI over all experience instances having *adj* as adjustment. The weights are given by the inverse of the distance between the adaptation contexts in the experience and $\mathbf{x}$. Adaptation contexts near $\mathbf{x}$ are assigned higher weights because it is assumed that their adjustments will have a similar effect in $\mathbf{x}$. The accuracy of the estimate depends on how close to $\mathbf{x}$ the adaptation contexts in the experience are.

Let $\mathbf{x}$ be an adaptation context, *adj* be a potential adjustment to assign to $\mathbf{x}$, and $\mathbf{y}_1,\ldots,\mathbf{y}_l$ be the experience instances having *adj* as adjustment. The estimated effect $\mathbf{E}(\mathbf{x},adj)$ of applying *adj* in $\mathbf{x}$ is given by

$$\mathbf{E}(\mathbf{x},adj) = \sum_{i=0}^{l} \frac{\delta_i}{d(\mathbf{x},\mathbf{y}_i)} \sum_{i=0}^{l} d(\mathbf{x},\mathbf{y}_i) \tag{3}$$

where δ_i is the difference in KPI between the initial and final state of experience instance i.

4.5 Ranking of Adaptations

A ranking of potential adjustments for an adaptation context $\mathbf{x}$ can be produced by combining distances and estimated effects, using the function

$$r_{\mathbf{x}}(\mathbf{y}) = \frac{d(\mathbf{x},\mathbf{y})}{\mathbf{E}(\mathbf{x}, adj(\mathbf{y}))} \tag{4}$$

where $adj(\mathbf{y})$ is the adjustment of experience instance $\mathbf{y}$. A list of adjustments $adj(\mathbf{y}_1), \ldots, adj(\mathbf{y}_m)$ can then be presented to the user, ordered by $r_{\mathbf{x}}(\mathbf{y})$.

4.6 Application of a Sequence of Adaptations

We have seen how to employ experience for providing decision support on which adjustment to apply in an adaptation context. This process can be repeated until the performance targets are reached. At each step, if experience related to the current context is available, the operator is presented with a list of recommended adjustments to choose from. The one that is applied determines the next adaptation context.

The process is illustrated in Fig. 2. The first adaptation context where decision support is sought is shown in (a). The initial state is S_1 and three adjustments a_1, a_2 and a_3 are provided, ordered by r_{S_1}. Adjustment a_1 is applied and the resulting state is S_2 (b). The recommended adjustments are now a_{21}, a_{22} and a_{23}. The operator can decide to either choose one of them or backtrack to the previous state and try out a different adjustment. The latter option is shown in (c).

Of course, an operator is free to make an adjustment that is not in the list of the recommended ones. This is common, for example, in the initial stage of the ramp-up phase, when little experience has been accumulated.

Whether or not adaptations are based on past experience, they produce new experience. Performance measurements of adaptations based on experience contribute to refining the experience related to that type of adaptation.

5 Architectural Design

The adaptation framework presented in this paper has been implemented within a larger system with a component-based software architecture, developed in the FRAME project.[1] One architectural requirement was to deploy the software

[1] "Fast Ramp-up and Adaptive Manufacturing Environment", funded by the European Commission's 7th Framework Programme.

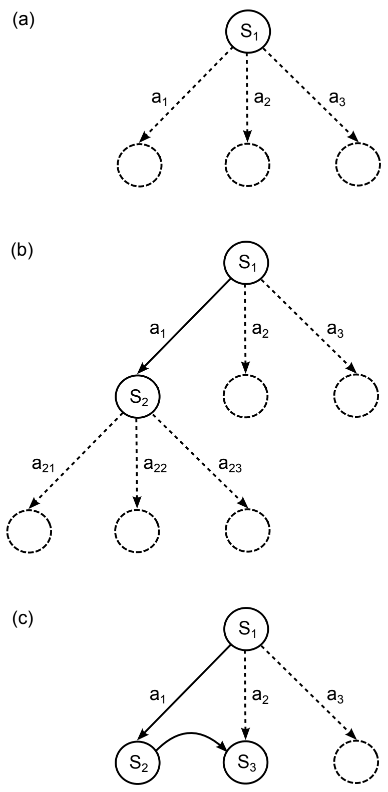

Fig. 2 Application of a sequence of adjustments: initial state (a); the first recommended adjustment a_1 is applied (b); the second recommended adjustment a_2 is applied after backtracking (c)

on assembly systems comprising several stations, in a distributed fashion. Each station can run a number of components for collecting or analysing data, acquiring experience or providing decision support on adaptation.

The architecture defines two levels (Fig. 3): a system level and a station level. Analysis and learning can occur at both levels. In addition to status events, the system level receives aggregated data (e.g. experience, batch statistics) from the stations. While learning at the station level is related to individual stations, learning at the system level is related to the entire assembly system and regards adaptations impacting some or all stations. One experience base is constructed for each station, as well as one for the whole system. In general, stations and system can be analysed using different KPIs. In addition, a recommended change to a station may not be recommended at the system level.

The communication infrastructure includes both asynchronous and synchronous communication mechanisms between components, which are employed in different scenarios. The event-based communication (Section 3) represents the asynchronous one and is based on the Data-Distribution Service for Real-Time Systems (DDS) (OMG, 2013). It involves sending data from a single component (publisher) to every component that has declared an interest in it (subscriber).

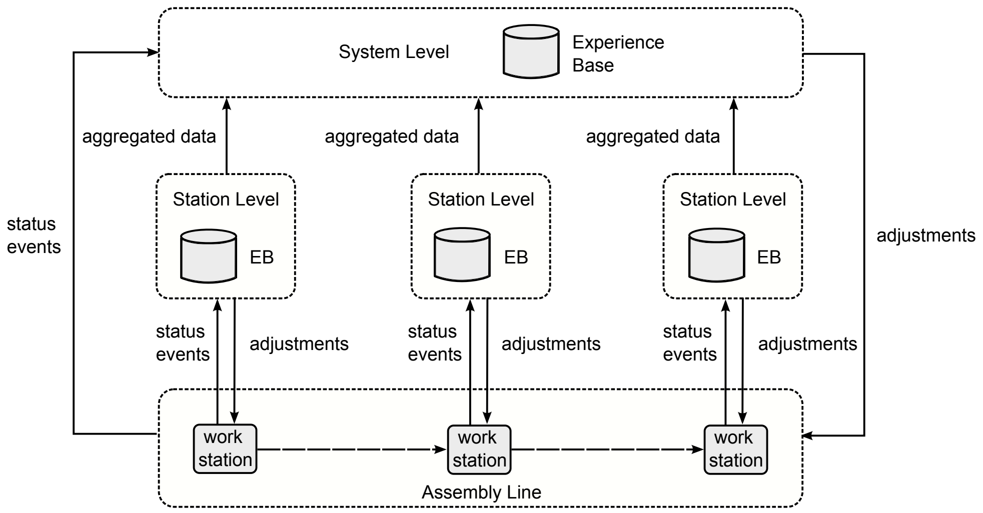

Fig. 3 2-level software architecture of the adaptation framework

Publishers and subscribers are decoupled and do not even need to know of the existence of each other. The publisher of a message marks it with a topic, while subscribers register to receive messages of certain topics. There are no central brokers transferring messages. All the communication takes place using a peer-to-peer mechanism. Scalability and performance are therefore higher because every message needs less hops to get to the receiver and there is no central bottleneck.

The DDS communication is used for broadcasting events because it is not known in advance which components will be interested in receiving them. It is not even known which components are running at a certain point. A station component does not normally subscribe to other stations' events and system events. Instead, system-level components may subscribe to both station and system events.

The Query/Response interface represents the synchronous communication and is implemented with web-services. It is a one-to-one communication between components, intended for queries with potentially large volumes of data (e.g. querying for all events generated in a specific period of time).

The Experience Recognition component, for example, subscribes to adjustment events, whereas the Automated Learning component queries the Experience Recognition for experience instances.

6 Experimental Evaluation

The proposed technique was evaluated using a modular assembly system for producing injection pens (Fig. 4). Two groups of 3 operators were asked to adapt the machine to the manufacture of a new product variant. The first group had no external support, while the second could use the software

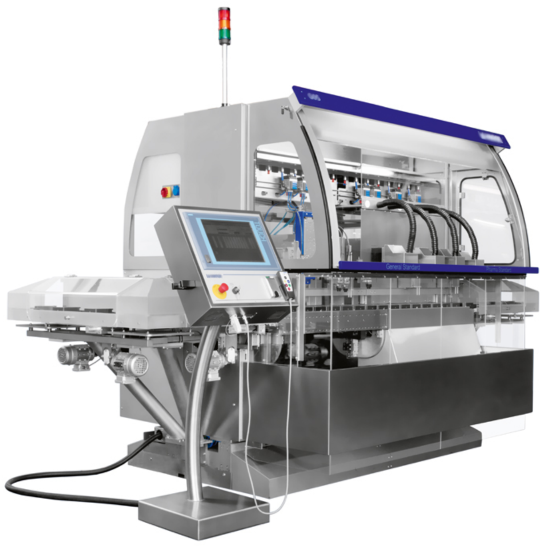

Fig. 4 Modular assembly system used in the experimental evaluation

framework implementing our technique and an experience base.

The adaptation process was carried out by each operator individually. All operators had comparable skills and experience. The initial process recipe was saved and after each adaptation it was restored, so that every operator could start from the same configuration and state. A KPI was calculated at each adaptation step made by an operator. The results of the two groups were compared.

An injection pen is composed of a body, a tank and a cap. The system comprises pick-and-place modules for inserting the tank in the body and putting the cap on top, as well as units for checking the presence and correct positioning of parts. The performance of the system is measured by verifying the quality of the final product. If a pen passes all checks then it is accepted, otherwise it is rejected. There is no rework on bad parts because it is not cost effective. A system KPI can be simply defined as the number of good parts out of the total number of parts produced in a batch or in a certain period of time.

After each assembly step, a checking unit verifies that the step has been performed correctly on the product. An individual assembly module, together with the associated checking device, can be considered a station in our 2-level architecture. The assembly system consists of 3 stations: one for placing the tank in the body, one for placing the cap on top and one for turning the pen upside down. A station KPI can then be defined as the number of good parts out of the total number of parts assembled by the station. By analysing stations separately it is possible to generate station-specific knowledge that can be reused when a station is installed on a different system.

The process recipe includes parameters such as the types of grippers and their pressure, the opening and closing angles, the pressure of cylinders, the program numbers of cameras and the type of pallet. The machine is cam-driven and its operational speed is fixed. This implies that cycle times cannot be altered and performance indicators based on cycle times are therefore not meaningful.

The assembly system was reconfigured for manufacturing a new injection pen composed of parts similar to the original ones but with slightly different dimensions. A number of mechanical adjustments to grippers and checking devices were made to enable the production of the new variant.

The system KPI that was calculated was the number of good parts divided by the total number of parts produced in a time period of 1 minute. Operators had access to the current KPI value all the time. Each operator was allowed to make 30 atomic adjustments. (Further adjustments would be required to achieve optimal performance.)

The first group adapted the machine without any external support and an experience base was built by capturing all the adjustments performed. Then, the second group repeated the process with the support of our software and the experience base. The operators of the second group followed the first recommended change 22, 20 and 24 times, respectively, and did not follow any of the recommended changes 3, 3 and 1 times, respectively.

Fig. 5 shows the KPI for the 3 operators that adapted the machine without external support. Fig. 6 shows the KPI for the 3 operators that adapted the machine with external support. Fig. 7 shows the average KPI for the two groups of operators. As it can be observed, Fig. 5 presents a number of steep rises and drops, which do not appear or are less marked in Fig. 6. This may be caused by a higher level of uncertainty of the operators of the first group about which adjustment to make. The average values in Fig. 7 indicate that the second group performed the adaptation faster and achieved a higher KPI at the end of the process.

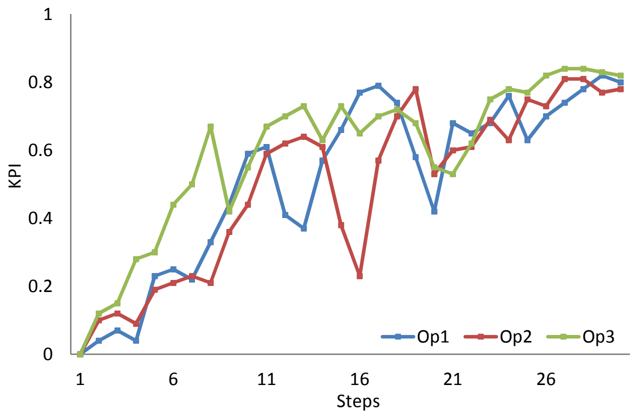

Fig. 5 KPI values for the operators that adapted the machine without external support

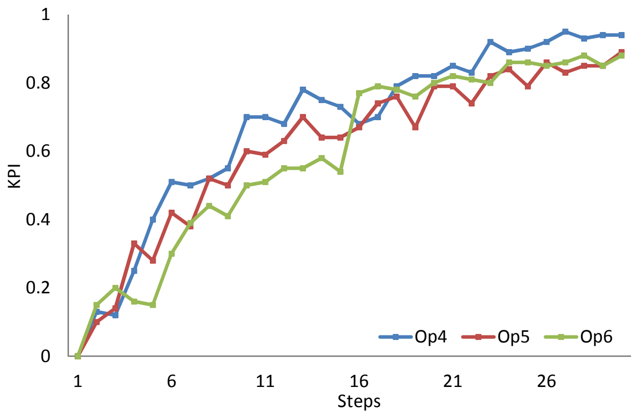

Fig. 6 KPI values for the operators that adapted the machine receiving recommendations on changes

An analysis of the impact of changes on individual stations was conducted using station KPIs. This revealed that the major difference in performance between the two groups was related to the first and second stations (those for inserting the tank in the body and placing the cap), while there was little difference in relation to the third station.

7 Discussion

Effectiveness and accuracy of recommendations depend on the adopted KPI and distance function. The estimated effect of an adaptation is based on the difference in KPI in the associate experience instances. The drawback of this approach is that, although state similarity is taken into account, an adaptation that caused a big improvement in a very bad state could be preferred to one that caused a slight improvement in a good state. To avoid this situation, absolute KPI values should be considered as well.

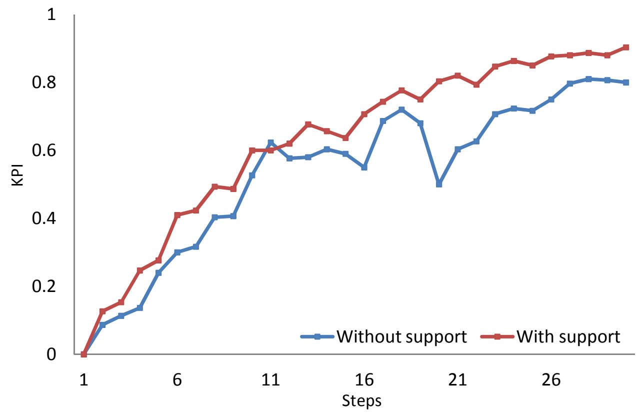

Fig. 7 Average KPI values for the two groups of operators (the one that received external support in the form of recommended changes, and the one that did not)

Structural adaptations that change the physical configuration of an assembly station can involve adding a new module or device, or replacing an existing one. Such adaptations are applicable only if the new module is compatible with the rest of the system. Similar adaptations contexts, according to the distance function, do not necessarily admit the same structural adaptations. Therefore, recommendations on adaptations provided exclusively on the basis of context similarity are not guaranteed to be applicable.

Having a specification of modules' physical requirements and compatibility conditions, for both installed and available modules, would allow to produce recommendations only on applicable structural adaptations. Analogously, if capability and logical requirements were also specified, the selection of structural adaptations could take them into account. This way, a module would be replaced only with one having equivalent capabilities and logical function in the system.

8 Conclusions

We have presented a technique for capturing information on adaptations through event generation and experience creation. Experience is used for providing decision support on adaptations. The problem of finding the best adjustment in a certain adaptation context is framed as a classification problem. A kNN algorithm is used for finding the most similar adaptation contexts in the experience. Adaptations are evaluated with a KPI and an estimate of the effect is calculated based on the experience. The architecture of the software framework implementing the technique has been outlined.

The effectiveness of the technique has been evaluated in a case study, in which an adaptation process was carried out on an assembly system to produce a new product variant. The experiment showed that the group that adapted the system receiving support achieved higher KPI values. Further experimentation with larger groups and more complex adaptations is required to draw firm conclusions.

Although the technique requires an appropriate representation of machine states and depends on the adopted KPI and distance function, it is of general application and straightforward use. Some improvements regarding the use of the KPI and future directions in relation to structural adaptations and physical/logical requirements have been discussed. There are also other aspects which are worth studying, such as the use of the technique to transfer knowledge between similar machines and operators with different skills, and the impact in terms of costs saved in the long run.

References

Arai T, Aiyama Y, Sugi M, Ota J (2001) Holonic assembly system with plug and produce. *Computers in Industry* 46:289–299

Barbosa B, Ferreira D (2013) Classification of multiple and single power quality disturbances using a decision tree-based approach. *Journal of Control, Automation and Electrical Systems* 24:638–648

Basri R, Hassner T, Zelnik-Manor L (2011) Approximate nearest subspace search. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33(2):266–278

Belongie S, Malik J, Puzicha J (2001) Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24:509–522

de Carvalho J, Coury D, Duque C, Paula B (2014) A new transmission line protection approach using cumulants and artificial neural networks. *Journal of Control, Automation and Electrical Systems* 25:237–251

ElMaraghy HA (2006) Flexible and reconfigurable manufacturing systems paradigms. *International Journal of Flexible Manufacturing Systems* 17:261–276

Fjällström S, Säfsten K, Harlin U, Stahre J (2009) Information enabling production ramp-up. *Journal of Manufacturing Technology Management* 20(2):178–196

Foguem BK, Coudert T, Béler C, Geneste L (2008) Knowledge formalization in experience feedback processes: An ontology-based approach. *Computers in Industry* 59:694–710

Frei R, Ferreira B, Serugendo GDM, Barata J (2009) An architecture for self-managing evolvable assembly systems. In: Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics

Jaber MY (ed) (2011) *Learning Curves: Theory, Models, and Applications*. CRC Press

Järvenpää E (2012) Capability-based adaptation of production systems in a changing environment. PhD thesis, Tampere University of Technology, Finland

Kim S (2003) Protein β-turn prediction using nearest neighbour method. *Bioinformatics* 20(1):40–44

Korena Y, Heiselb U, Jovanec F, Moriwakid T, Pritschowb G, Ulsoya G, Brussel HV (1999) Reconfigurable manufacturing systems. *CIRP Annals - Manufacturing Technology* 48:527–540

Lazzaretti A, Ferreira V, Neto H, Riella R, Omori J (2013) Autonomous neural models for the classification of events in power distribution networks. *Journal of Control, Automation and Electrical Systems* 24:612–622

Oates RF, Scrimieri D, Ratchev S (2012) Accelerated ramp-up of assembly systems through self-learning. In: Proceedings of the 6th IFIP WG 5.5 International Precision Assembly Seminar (IPAS 2012), Springer, pp 175–182

OMG (2013) Data-distribution service for real-time systems. http://portals.omg.org/dds/, accessed on 31 January 2014

Scrimieri D, Ratchev S (2013) Capture and application of adaptation knowledge on assembly stations. In: Proceedings of the 11th IFAC Workshop on Intelligent Manufacturing Systems, pp 87–92

Scrimieri D, Oates R, Ratchev S (2013) Learning and reuse of engineering ramp-up strategies for modular assembly systems. *Journal of Intelligent Manufacturing* DOI 10.1007/s10845-013-0839-6

Surbier L (2010) Problem and interface characterization during ramp-up in the low volume industry. PhD thesis, Institut polytechnique de Grenoble, France

Terwiesch C, Xu Y (2004) The copy-exactly ramp-up strategy: trading-off learning with process change. *IEEE Transactions On Engineering Management* 51(1):70–84

Westkämper E (2006) Factory transformability: Adapting the structures of manufacturing. In: Daschenko A (ed) Reconfigurable Manufacturing Systems and Transformable Factories, Springer Berlin / Heidelberg, pp 371–381

Wilson DR, Martinez TR (1996) Instance-based learning with genetically derived attribute weights. In: Proceedings of the International Conference on Artificial Intelligence, Expert Systems and Neural Networks (AIE'96), pp 11–14

Wilson DR, Martinez TR (1997) Improved heterogeneous distance functions. *Journal of Artificial Intelligence Research* 6:1–34