Available online at www.sciencedirect.com

SCIENCE DIRECT®

ELSEVIER

Int. J. Production Economics 100 (2006) 87–100

international journal of production economics

www.elsevier.com/locate/ijpe

Operation management in reconfigurable manufacturing systems: Reconfiguration for error handling[☆]

Manfredi Bruccoleri[a,*], Zbigniew J. Pasek[b], Yoram Koren[b]

[a]*Dipartimento di Tecnologia Meccanica, Produzione e Ingegneria Gestionale, Faculty of Engineering, Università di Palermo, Viale delle Scienze, 90128 Palermo, Italy*

[b]*NSF Engineering Research Center for Reconfigurable Machining Systems, University of Michigan, Ann Arbor, MI, USA*

Received 8 December 2003; accepted 13 October 2004
Available online 8 December 2004

Abstract

Reconfigurable manufacturing systems offer quick adjustment of production capacity and functionality in response to unpredictable market changes as being systems designed at the outset for rapid change in system configuration, its machines and controls. During the production process, out-of-ordinary events occur dynamically and unpredictably both at the system (machine breakdowns, change in job's priorities, etc.) and at the cell level (tool failures, robot collisions, etc.). Such exceptions interrupt the production process by causing errors in the schedule plan (system level) or in the task plan (cell level). Error handling is the policy meant for reacting to errors caused by the occurrence of out-of-ordinary events. The reconfiguration ability turns out to be the new technological factor enabling new strategies to handle out-of-ordinary events of the production process. Both economic and performance aspects need to be considered in order to make a decision in support of particular error handling policies such as using reconfiguration. This paper, starting from a simulation case study, highlights advantage of using *reconfiguration for error handling*. Authors propose an object-oriented high-level control structure for real-time error handling, which integrates the new reconfiguration for error handling technology with the existing reactive scheduling system.
© 2004 Elsevier B.V. All rights reserved.

Keywords: Error handling; Object-oriented methods; Reconfigurable manufacturing

1. Introduction

The ability to launch new product models quickly, the potential for rapid alteration of manufacturing system capacity and the fast integration ability of new process technologies into existing systems are the emerging requirements for the contemporary and future production

[☆]This research was supported by the grant of National Science Foundation (Grant #EEC-9529125).

[*]Corresponding author. Tel.: +39 091 6657036; fax: +39 091 6657039.
E-mail address: manbru@dtpm.unipa.it (M. Bruccoleri).

0925-5273/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.ijpe.2004.10.009

facilities. Reconfigurable manufacturing systems (RMSs) are at the moment widely considered as one of the promising key technologies to enable responsiveness in the new production era known as mass customization. It is true that the flexibility offered by flexible manufacturing systems (FMSs) certainly allows to manufacture a variety of products in the same systems, but it is not the only key to success. In order to remain competitive under unpredictable and rapid changing market conditions, flexibility must be coupled with responsiveness and cost efficiency, as well as high reliability, scalability, and ability for easy software/hardware upgrades ability (Mehrabi et al., 2000). RMSs offer such features allowing quick adjustment of production capacity and functionality by rearranging or changing its modular components.

Koren et al. (1999) define RMS as a manufacturing system designed at the outset for rapid changes in structure, as well as in hardware and software components, in order to quickly adjust production capacity and functionality within a part family in response to sudden changes in market or in regulatory requirements. An RMS is basically a mix of CNC machines, dedicated machines, and reconfigurable machine tools (RMTs). RMTs are modular machines with a flexible structure that allows changes of its resources (e.g. adding a new axis), equipped with reconfigurable controllers integrated in open-architecture environment. An RMS can be easily reconfigured at a system level (e.g. changing the layout configuration), machine level (e.g. adding a new spindle), and control level (e.g. integrating a new software module) (Koren et al., 1999).

An exception can be thought of as a difference between the actual and the expected state of the production system. Machine breakdowns, changes in job priorities, dynamic introduction of new jobs, order cancellations, increases in job arrival rates, changes in the mix of parts, and reworks due to quality problem, are all examples of exceptions. Exception occurrence is usually unpredictable. However, predictable events, such as planned preventive maintenance, can also be considered exceptions as they interrupt the production process. Unless some strategy to deal with the exception is implemented, the production system is forced to stop its operation whenever an exception occurs. Otherwise, damages to the system itself or errors in the product may occur. Exception handling is the policy meant for countering unwanted effects of exceptions and for recovering from errors caused by exception occurrences.

As the literature review shows, a considerable effort has been made to investigate the field of exception handling for the scheduling or task planning process. Most frequently, the adopted approach to exception handling is based on strategies which are enabled by the system routing flexibility. The issue treated in this paper is, instead, to understand how the reconfigurability feature of RMSs could be also used together with the routing flexibility in the exception-handling process. This would make the RMSs surely more cost effective. In RMS, however, reactive scheduling decisions, task planning recoveries, and error handling policies are naturally more complex since the use of hardware reconfiguration in these systems for handling exceptions is secondary. In fact, the reconfiguration ability of these systems could be used in exceptional cases, as production process errors, even though it cannot be considered a source of flexibility for ordinary operations.

The research presented in this paper explores such potential directions and its focus is on understanding whether the reconfiguration ability of RMSs can be used and coupled with the routing flexibility in the error handling process, further increasing its cost-effectiveness. In order to make decisions regarding *reconfiguration for error handling*, economic and performance considerations have to be carried out and a real-time decision support system is needed. Such system has to be coupled with the existing scheduling system (scheduler) at the system level and ultimately with the sequence controller (task planner) at the cell level. For these reasons, a modular and object-oriented (OO) architecture is needed for operation management issues such as error handling in RMS (Bruccoleri et al., 2003b).

The proposed reconfiguration for error handling at the system level is addressed in Section 2, while Section 3 addresses the same problem at the cell level by providing a literature review on error recovery strategies in robotized cells, stating the explored problem, analyzing the error handling issue in a manufacturing transfer line case study. The proposed control model is shown in Section 4. Conclusions and further developments are reported in Section 5.

2. Error handling in scheduling systems

2.1. Traditional approaches

The decisions concerning how to deal with exceptions, i.e. which strategy to implement, depend on the final goal of the operation management and involve many economical and production variables. Also, they depend on the manufacturing type of environment (dedicated, flexible, etc.), and the advantages arising from using a certain exception handling policy vary from one production system to another.

When an exception occurs, the whole or a part of the schedule plan is concerned. Thus, the development of a preventive off-line scheduling or that of implementing a real-time decision support system to react to exceptions is strictly required. Three main ways to develop reactive scheduling systems for dynamic and uncertain production systems can be found in the literature. They are: predictable scheduling, pure reactive scheduling, and predictive–reactive scheduling.

Predictable scheduling approaches focus on the development of predictive schedules that can absorb disruptions without affecting planned external activities while maintaining high shop-floor performances. The preventive off-line scheduling is necessary to provide the coordination to many other production activities such as set-ups, maintenance, personnel management, material procurement, shipping, etc. The preventive schedule can be obtained, for instance, by inserting additional idle time into the schedule to adsorb the impacts of breakdowns (O'Donovan et al., 1999).

In pure reactive scheduling approaches, no preventive schedule is generated in advance, and decisions are made locally in real-time to deploy corrective mechanisms in order to maintain the stability of existing schedules and provide quick solutions in response to the dynamic and uncertain exception described above. Dynamic dispatching rules' selection, based on jobs, machines and system status is a frequently used pure reactive approach (Park et al., 1997).

Finally, the predictive–reactive scheduling system includes both of the elements: a long-term preventive scheduling system and a reactive scheduling system (Sabuncuoglu and Bayiz, 2000). Artificial intelligence techniques (Belz and Mertens, 1996), agent-based systems (Adacher et al., 2000; Brandimarte et al., 2000), holonic systems (Gou et al., 1998), simulation techniques (Heng et al., 2000), beam search (Unal et al., 1997), knowledge-based systems (Dutta, 1990; Mehta and Uzsoy, 1997), heuristic algorithms (Jain and Elmaraghy, 1997), dynamic selection of dispatching rules (Holthaus, 1999; Piramuthu et al., 2000; Seifert and Morito, 2001), are some of the most utilized methods for developing reactive scheduling systems.

A literature survey on reacting scheduling papers from 1980 to 1990 can be found in Sabuncuoglu and Bayiz (2000), while from 1990 to 2000 in Bruccoleri et al. (2003a).

2.2. Reconfiguration for error handling

One very important observation stemming from the literature analysis is that reactive scheduling systems are designed at the outset to respond to exceptions by using the operational flexibility of the system. FMSs and job shop systems generally allow flexibility in the routing of the parts through the system. On the contrary, dedicated manufacturing lines, which consist of a number of dedicated machines performing single operations sequentially, and RMSs (if they do not include any flexible machine and basically are made up of some rigid machining station and a number of RMTs) do not allow any routing flexibility unless there are more machines performing the same operation. Nonetheless, although the reconfiguration ability cannot be thought as routing flexibility and used for ordinary scheduling operations, it can be rather employed as an operational tool in special cases, such as out-of-ordinary events.

Exploring this idea, Bruccoleri and Pasek (2002), analyzing a number of case studies, showed and illustrated that, in particular conditions, at the operation management level (error handling), the reconfiguration can be a crucial technology in manufacturing systems where the routing flexibility is not allowed. Also, when coupled with the existing reactive scheduling system, it can also enhance performance of systems that already offer operational flexibility.

Starting from a reconfigurable production system (consisting of one RMT and one traditional rigid machine tool), Amico et al. (2001) developed an intelligent centralized controller, based on fuzzy systems, which handles exceptions by using reconfiguration. For that simple manufacturing system, the authors demonstrated that reconfiguration for error handling brings advantages in the production system productivity. A more complex and distributed controller, based on multi-agent systems, has been proposed by Bruccoleri et al. (2002) for testing the same operation management policy in a more realistic manufacturing system composed by several machine tools, some of which are RMT. Once again, they demonstrated that, under specific constructive conditions (e.g. the reconfiguration time is minor than the time to repair the broken machine), reconfiguration for error handling is an effective operation strategy in terms of production system performance.

3. Error handling in manufacturing cells

3.1. Literature review

As was shown in the previous section, from a system-level point of view, the error handling issue is mainly concerned with the production scheduling system and deals with unexpected events like machine breakdowns, changes in job priorities, and so forth, which can be identified by a system-level controller. Besides these kinds of error, out-of-ordinary events at the cell level have also to be considered.

A manufacturing automated/ robotized cell can be defined as an arrangement of manufacturing equipments (robots, NC machines, material handling systems, sensors, PLCs, cell controllers, etc.) grouped to effectively process a set of products of the same part family (Jang et al., 1997). From the operation management perspective, the control of an automated cell basically is implemented in the form of a sequential control program. During the production process, no routing flexibility is needed, nor put into operation. It can be thus stated that, from this point of view, a manufacturing cell, a manufacturing line, or an assembly line can be treated in the same way. A reactive task planning method is strictly required to respond to out-of-ordinary events.

Exceptions in an automated/semi-automated robotized cell can be classified as unexpected events (Borchelt and Thorson, 1997) (like time-out on expected process report or occurrence of unexpected reports), assembly errors (Kao, 1995; Najjari and Steiner, 1997) (like positioning errors), or unpredictable failures (Wu, 1999) (such as resources' out of orders, cameras' failures, equipment failures, tool breakages, human errors, material handling problems, collisions, obstructions and handling failures). In all of these cases, as for the system-level exceptions, exceptions can be thought as differences between the actual and the expected state of the system.

In the literature, error handling issues at the production cell level have been largely investigated because statistics show that in a robotic environment the code for handling sensors often takes up 80% of the robot's task program and, even in a simple production process, about 90% of the system coding is devoted in automatic error recovery (Hasegawa et al., 1990); also, operational faults associated with PLC control processes occur most often (about 70%) among all kind of faults, and when operational faults occur, about 80% of downtime is spent in locating its source and 20% is spent on the repair (Hu et al., 1999).

Since it can be stated that faults may be due to the occurrence of events, such as system deterioration or built-in imperfections, whereas errors are the manifestation of these faults (i.e. detected discrepancies between the actual and the expected state of the system) and may be proclaimed failures depending on their severity (Kokkinaki and Valavanis, 1996), the error handling in an automated production cell could be decomposed in the following phases (Toguyeni et al., 1996):

- fault DETECTION and IDENTIFICATION,
- error DIAGNOSIS (identify the components which are responsible for the system degradation) and PROGNOSIS (identify the future degradation consequences),
- error or failure RECOVERY.

Almost all the approaches, which can be found in the literature, employ knowledge-based techniques to implement intelligent and real-time error diagnosis and recovery (Borchelt and Thorson, 1997). The knowledge-based system is often associated with AI algorithms (Klein et al., 1996) or expert systems for training and learning (Lopes and Camarinha-Matos, 1995), and they are then integrated with other OO software control modules (Kopacek et al., 1999), or with other analysis tools like state diagrams (Kokkinaki and Valavanis, 1996) or Petri nets (Jeng and Mu Der, 1997). Also, depending on the control system implemented in the automatic robotized manufacturing/assembling cell, the literature proposes different hierarchical control architectures where to integrate the error-handling module.

The task planning is usually implemented in a PLC, but it can be directly run by a PC. In most cases, the planner is the sequence controller that controls the I/O variables synchronization by using simple algorithms or programs. The error handling functions are usually integrated with the regular operation program, i.e. the task plan. Thus, at this level, the error handling is quite limited and, anyhow, complex error handling functions require very complex programs.

Several modeling tools can be used to design reactive task planners, i.e. sequence and synchronization controllers with error handling abilities. In a chronological order of appearance in the literature, ladder diagrams (Hasegawa et al., 1990), Petri nets diagrams (Hasegawa et al., 1990) and finite-state machine diagrams (Borchelt and Thorson, 1997) are the most used modeling tools.

A classification of some research papers is shown in Table 1. They are basically classified according to the error handling phases they focus on and the approach they propose. It has to be noticed that most of them are mainly focused on robotized cell control software architecture, so the error handling issues is secondarily treated.

3.2. Reconfiguration for error handling: A case study

This section presents a case study intended to show and demonstrate the proposed operation management methodology for handling out-of-ordinary events, even at the cell level. This approach is based, once again, on the reconfiguration ability of RMSs. Notice that in the cell level, the reconfiguration for error handling strategy could reveal to be even more crucial because at this level the routing flexibility is not allowed or is totally missed, as in robotized cells or manufacturing lines, where the task sequences are usually very rigid.

In the presented case study, the error handling function has been implemented in a FisherTechnik® manufacturing line toy (Fig. 1) installed in the Engineering Research Center for RMS of the University of Michigan. The manufacturing system toy is a tightly coupled transfer line composed of three stations (a drilling station, a vertical milling station and a horizontal milling station) with no buffers between them. The system processes only one type of part, flowing through the system on a conveyor. All the stations are subject to potential failures.

The case study analysis has been performed by, first, implementing the error handling based on a traditional control policy in which the error handling function is integrated with the ordinary task program (phase 1). This program runs on the PC-based controller written in C++ programming language. Then, a simulation model of the toy itself has been employed in order to contextualize the manufacturing system toy in a realistic manufacturing scenario and implement the proposed error handling strategy based on reconfiguration (phase 2). The simulation model has been developed in the Arena® discrete-event simulation environment (by Rockwell Software, Inc.).

Table 1
Classification of papers on error handling in manufacturing cells

Year	Title and authors	Journal/proceeding	Production environment	Error	Error handling	Approach	Modeling tool
1990	Modeling of exception handling in manufacturing cell control and its application to PLC programming. Hasegawa, M., Takata, M., Temmyo, T., Matsuka, H	Proc 1990 IEEE Int Conf Rob Autom. Publ by IEEE, Computer Society, Los Alamitos, CA, USA. p. 14–519	Flexible manufacturing cell (general)	Exceptional operations (general)	Error recovery	Layered Petri net with priority control	PN
1995	Machine learning approach to error detection and recovery in assembly. Lopes, L.S, Camarinha-Matos, L.M.	IEEE International Conference on Intelligent Robots and Systems v 3 1995. IEEE, Piscataway, NJ, USA. p. 197–203	Robotized assembling system	Collisions, obstructions and handling	Error detection and recovery	SKIL algorithm for knowledge-based system and machine learning	Prolog for implementing the planner, SKIL algorithm for failure classification
1995	Optimal recovery strategies for manufacturing systems. Kao, Jih-Forg	European Journal of Operational Research, v 80 n 2 Jan 19 1995. p. 252–263	Automated single-server assembly system	Detectable faults (part imprecision)	Error recovery	Optimization of throughput and profit for determining the recovery action	State transition diagrams and semi-Markovian model
1996	Error specification, monitoring and recovery in computer-integrated manufacturing: an analytic approach. Kokkinaki, A.I., Valavanis, K.P.	IEE Proceedings: Control Theory and Applications, v 143 n 6 Nov 1996. p. 499–508	Automated three-robot system	Resources' out of orders and cameras' failures,	Error specification language, error monitoring, error recovery	Formal language for error specification and classification; error recovery based on a continuous mapping of the error severity values onto agent acting	State transition
1996	Automatic synthesis of control programs in polynomial time for an assembly line. Klein, I., Jonsson, P., Backstrom, C.	Proceedings of the IEEE Conference on Decision and Control v 2 1996, p. 1749–1754	LEGO car factory	General	Error recovery	Starting from an off-line development of the process model (stored in a database) and the actual state of the system a polynomial time algorithm generates the on-line control strategy.	GRAFCET charts automatically translated into PLC code using a commercial PLC compiler

1996	Framework to design a distributed diagnosis in FMS Toguyeni, A.K.A. Craye, E., Gentina, J.C.	Proceedings of the IEEE International Conference on Systems, Man and Cybernetics v 4 1996. IEEE, Piscataway, NJ, USA,96CH35929. p. 2774–2779	FMS (discrete event system)	Symptom I (time out on expected process report), Symptom II (occurrence of unexpected reports)	On-line error diagnosis: localization, identification, prognosis	Local diagnosis (by CTS rule formalism) for localization and overall diagnosis for identification and prognosis	FG (Functional Graph) and CTS (Causal Temporal Signatures) formalism
1997	Integrated sensor-based control system for a flexible assembly. Najjari, H., Steiner, S.J	Mechatronics, v 7 n 3 Apr 1997. p. 231–262	Flexible robotic assembly cell	Assembly errors (positioning errors)	Error detection and recovery	When an error starts to occur, detection sensors are installed and recovering programs are written	Knowledge base system
1997	Petri nets for modeling automated manufacturing systems with error recovery Jeng, Mu Der	IEEE Transactions on Robotics and Automation, v 13 n 5 Oct 1997. p. 752–760	Automated manufacturing system	General	Error recovery	Liveness-checking algorithm to check the net reversibility	Synthesized PN
1997	Toward reusable hierarchical cell control software. Borchelt, R.D. Thorson, J.	International Journal of Production Research, v 35 n 2 Feb 1997. p. 577–594	Robotic work cell	Operational errors	Error recovery	A guiding hand controller based on AI techniques assist the dynamic cell in handling unexpected operational errors	State diagrams for modeling the dynamic cell system and expert system (AI based) for the error recovery
1997	Application of design and control tools in a multirobot cell. Jang, J., Koo, P., H., Nof, S.,	Computers & Industrial Engineering, v 32 n 1 Jan 1997. p. 89–100	Multi-robot cell	Unexpected events	Error recovery	The operation sequence control is based on synchronous and asynchronous sequence controls. The error handling by using CTRERR.	PN for avoid deadlocks of the asynchronous sequences
1999	A Knowledge-based real-time diagnostic system for PLC controlled manufacturing systems. Hu, W., Schroeder, M., Starr, A.G.	Proceedings of the IEEE International Conference on Systems, Man and Cybernetics v 4 1999. IEEE, USA. p. IV-499—IV-504	General manufacturing system	General	Faults diagnosis: fault positions, causes, and corrective actions	Knowledge based diagnosis system with artificial knowledge acquisition based on logic programming	Knowledge-base system

Table 1. (*continued*)

Year	Title and authors	Journal/proceeding	Production environment	Error	Error handling	Approach	Modeling tool
1999	A modular control system for flexible robotized manufacturing cells Kopacek, P., Kronreif, G., Probst, R.	Robotica, v 17 n pt 1 Jan–Feb 1999. p. 23–32	Robotized manufacturing cell	Warnings or general errors	Fault recovery	The sequence controller handle the errors by sending an acoustic error as well as an appropriate message window and the assembling process stops	Object-oriented architecture of the control system
1999	Methodology of generating recovery procedures in a robotic cell. Wu, Hsien-Jung	Proceedings—IEEE International Conference on Robotics and Automation, v 1 1999. p. 799–804	Robotic cell	Equipment failures, tool breakage, human error, material handling problems	Error recovery	Preparation (algorithms for the generation of an extended task list with recovery procedures), generation (algorithms for connecting the error recoveries to the controller)	Knowledge base, algorithmic procedures

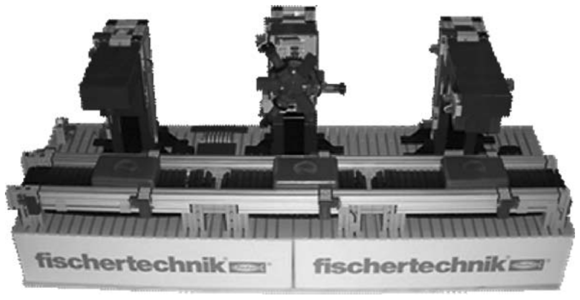

Fig. 1. The manufacturing system toy.

3.2.1. Phase 1: Traditional error handling implementation

As already mentioned, in the first phase, some error handling strategies have been added and implemented in the task plan software program, which control the task plan of the manufacturing cell operation. Actually, the real C++ code has been directly obtained by using a home made software tool which automatically generates the C++ code starting from a particular state transition diagram designed by the programmer. The specific state transition diagram consists of states, associated with actions (OUTPUT or engine variables), and transitions, corresponding to conditions (INPUT or sensor variables). Fig. 2 reports an example of the state transition diagram designed for a generic control program. It also includes the error handling procedures. Three examples of simple error handling policies have been implemented, namely: if the drilling machine fails, the controller stops the system and just waits for the operator (Error #1); if the tool of the vertical milling machine breaks down, then the controller stops the system, resets the vertical milling machine, rotates the tool axes and changes the mill, and restarts the operation (Error #2); if the horizontal milling machine breaks down while milling the part, the controller stops the system, resets the machine, and waits for the operator (Error #3).

In this traditional approach, the error handling function is rather limited and strongly depends on the flexibility of the manufacturing cell. In general, complex error handling functions require very complex control programs and, thus, complex state transition diagrams.

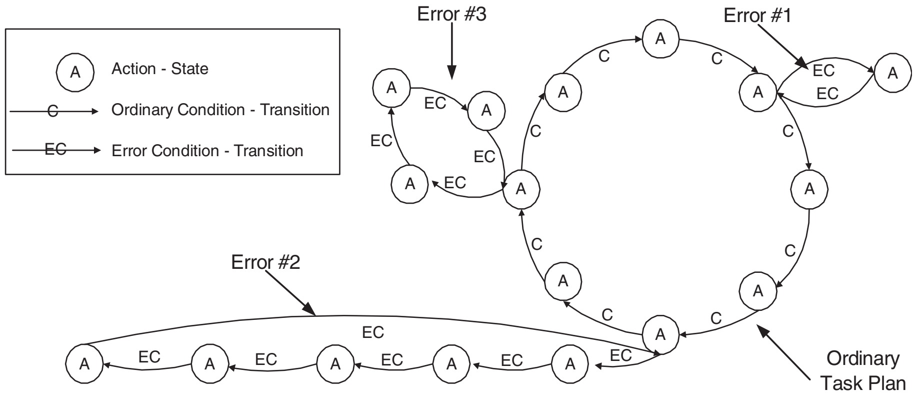

Fig. 2. A state transition diagram for reactive task planning.

In the second phase of the analysis, it will be demonstrated how the proposed error handling approach based on reconfiguration can overcome these limitations. This phase is described in the following sections.

3.2.2. Phase 2: Error handling by means of reconfiguration

The simulated production system consists of two different and independent subproduction systems (Fig. 3). Namely:

- *System A* (*FisherTechnik*®): tightly coupled transfer line composed of three stations without any buffers between them and designed to manufacture only one type of parts. All the stations are subject to failure.
- *System B*: reconfigurable system composed of a mix of flexible, reconfigurable, and dedicated machines and designed at the outset to manufacture many part types from the same part family.

The basic assumption is that System B contains a reconfigurable machine (RMS_HM) that, if reconfigured, can perform the operation ordinarily performed by the H_Milling Station of the System A.

Once again, authors want to test the strategy of reconfiguring RMS_HM to handle the H_Milling Station breakdowns when these occur.

For obvious reasons, the performance of the two systems will behave in an opposite way with respect to each other, if a machine of the System B is used to replace temporarily another associated machine of System A. Specifically, the performance of System A will improve, thanks to a new available resource (RMS_HM), while the performance of System B will worsen, because one of its resources is now shared with the other system.

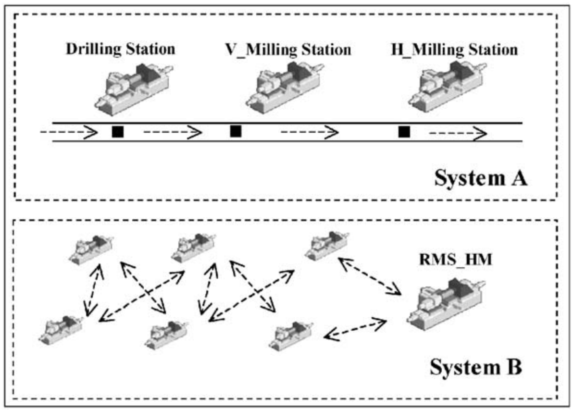

Fig. 3. Imaginary production system configuration.

As a consequence, economic considerations would reveal to be necessary in order to make the final decision on the exception handling policy to adopt. That is, a comparison between the costs of adopting one strategy over another is required in order to make the selection. In the following, the proposed economic model for this case study is presented and the simulation results are shown.

3.2.3. Economic model

When the H_Milling Station failures occur, System A performance worsens because the system is forced to reduce its throughput due to unavailability of the above-mentioned station. The depreciation of productivity is well represented by the increase of the mean cycle time $\Delta \mathrm{CT}_f$ of the parts.

If an exception handling policy is implemented (in this case, the reconfiguration of the RMS_HM), the productivity of the System A improves, but is still inferior to that in case of no failure. Summing up:

$$\begin{aligned}
&\text{No Failure} :\rightarrow \text{cycle time} = \mathrm{CT},\\
&\text{No Reconfiguration} :\rightarrow \text{cycle time}\\
&\quad = \mathrm{CT}_1 = \mathrm{CT} + \Delta\mathrm{CT}_f,\\
&\text{Yes Reconfiguration} :\rightarrow \text{cycle time}\\
&\quad = \mathrm{CT}_2 = \mathrm{CT} + \Delta\mathrm{CT}_r,\\
&\mathrm{CT}_1 = \mathrm{CT}_2 + \Delta\mathrm{CT},\\
&\text{where } \Delta\mathrm{CT} = +\Delta\mathrm{CT}_f - \Delta\mathrm{CT}_r.
\end{aligned}$$

The increase of cycle time (ΔCT) has a cost, and then it can be stated that the cost of "not using reconfiguration versus using reconfiguration" is

$$\begin{aligned}\text{Cost(Not Rec vs. Yes Rec)} &= \text{Cost}(\mathrm{CT}_1 - \mathrm{CT}_2)\\ &= \text{Cost}(\Delta\mathrm{CT}). \qquad (1)\end{aligned}$$

On the other hand, using the RMS_HM for replacing the broken H_Milling Station, while improving System A performance, obviously deteriorates the performance of the System B. How the performance decreases varies dynamically and depends on how many part types the System B is processing and which of those are processed by the RMS_HM.

It can be said, though, that the decreasing of the System B performance (ΔP_r), due to the reconfiguration of the RMS_HM for handling the exceptions of System A, depends on the interval of time that this machine stays busy for replacing the H_Milling Station. So,

$$\begin{aligned}\Delta P_r &= f(\text{time percent RMS is busy for System A})\\ &= f(\%b).\end{aligned}$$

Also in this case, the decrease of the System B performance (ΔP_r) has a cost and, analogously to the economic model of System A, it can be stated that the cost of "using reconfiguration versus not using reconfiguration" is

$$\text{Cost(Yes Rec vs. Not Rec)} = \text{Cost}(\Delta P_r). \qquad (2)$$

Summing up, the reconfiguration strategy for error handling will be adopted if the following condition is true:

$$\text{Cost}(\Delta P_r) < \text{Cost}(\Delta\mathrm{CT}). \qquad (3)$$

Assuming that k ($0<k<1$), costA, costB are constant parameters and:

- $\Delta P_r = k \times \%b$,
- $\text{Cost}(\Delta P_r) = \text{costB} \times \Delta P_r$,
- $\text{Cost}(\Delta\mathrm{CT}) = \text{costA} \times \Delta\%\mathrm{CT}$,

where $\Delta\%\mathrm{CT} = (\mathrm{CT}_1 - \mathrm{CT}_2)/\mathrm{CT}_2$, condition (3) becomes

$$\Delta\%\mathrm{CT} > k \times \%b \times \frac{\text{cost B}}{\text{cost A}}. \qquad (4)$$

3.2.4. Simulation data and results

The presented model has been tested in a simulation environment in which the imaginary manufacturing system has been properly modeled and the following data have been assumed:

- the two systems have been observed for 10,000 minutes (about one production month);
- the processing times of all the manufacturing resources are triangularly distributed with mode mpt, minimum ($mpt-\Delta$) and maximum ($mpt+\Delta$), $mpt = 20$ minutes;
- all the manufacturing resources are subject to failure and the failures occur according to exponential distributions with MTBF = $20mpt$;
- the repairing times are crisp and equal to $ttr = 5mpt$;
- the reconfiguration time of the reconfigurable machine is crisp and equal to $T_{\text{rec}} = 3mpt$;
- the transfer times are exponentially distributed with mean $mtt = 10$ minutes;
- the load/unload times from the input warehouse and to the output depart station are equal to $t_{l/u} = \frac{1}{2} \cdot mtt$;
- cost A = \$ 80,000;
- cost B = \$ 100,000;
- $k = 0.3$.

Table 2 summarizes the simulation results in terms of volume produced and mean cycle time (for System A) before and after having implemented the error handling strategy. Note that the performance results were actually fully expected, according to the considerations on System A made in the previous subsection. Because of the random input and in order to guarantee a statistical validity of the results, for each run, the number of executed replications guarantees, for the output performance measures, that the length of confidence intervals (95% level) of the mean among replications is lower than 10% of the mean itself. Also, notice that the simulation results depend on the supposed scenario (input) and are obviously not general. Finally, notice that the reconfiguration time T_{rec} has been assumed to be bigger than the time to repair *ttr*. This choice, which strictly conditions the obtained results, avoids most of the questions already discussed in Bruccoleri et al. (2003a).

Table 3, on the other hand, shows the performance depreciation and costs incurred, respectively, in the case of "not using reconfiguration versus using reconfiguration" and vice versa, when both the systems are taken into consideration.

From a quick analysis of the results, it emerges that using the reconfiguration of the RMS_HM (System B) to replace the broken H_Milling Station (System A) does not bring global economic advantages. However, this result strongly depends on the specific value of *k*, which is the measure of how much the percent of time (%*b*)—in which the reconfigurable machine is busy to replace the broken machine of System A—influences the global performance of the System B itself. It can be easily computed that for $k_0 < 0.217$ the proposed exception handling strategy becomes economically viable.

Table 2
Case study–performance results

	Volume produced	CT
No Failure	252	39.68
No Reconfiguration	233	42.92
Yes Reconfiguration	244	40.98

Table 3
Case study–global economic results

	Δ%CT	%*b*	Total cost
Not Rec vs. Yes Rec	4.72%	—	$3776.82
Yes Rec vs. Not Rec	—	17.4%	$5200.00

4. Error handling control model

The error handling function in scheduling systems or task planners becomes more complex in RMSs due to the possibility of using reconfiguration to deal with out-of-ordinary events, primarily due to the increase in the number of feasible error handling strategies. In this section, an OO control architecture for the error handling is proposed. Fig. 4 shows the class diagrams of this model through the unified modeling language (UML) notation. The UML is the standard universal language, approved by the Object Management Group, for representing (i.e. specifying, building, and documenting) OO software system (Stevens and Pooley, 2000). Among competing OO analysis and design tools, the UML was selected because it defines a meta-model-based graphical notation for OO analysis and design, embracing all the features of OO paradigm such as reusability, representational versatility, inheritance property, and rapid prototyping.

In Fig. 4 the objects and their classes directly involved in the control process are shown. These classes include the machine tools, the task to be performed (i.e. the job), the scheduler or task planner, the sensor, the error handler, the reconfiguration controller, and the decision support system (DSS). The DSS class is responsible for making decisions about which error handling strategy to select, basing on performance and economic considerations, as highlighted in previous section. In such a class diagram the relationships among classes (aggregation, generalization/specification, and association) are shown as well. For the sake of brevity and clarity, in this paper the features of UML diagrams are not explained in detail; for a detailed description of UML diagrams and their application to manufacturing systems control modeling the reader can refer to Stevens and Pooley (2000).

A detailed description of the control process for error handling is described in an UML sequence diagram (Fig. 5), where the classes, already defined in the previous class diagram, are instantiated into objects, which send messages to each other. The control process is hence accomplished by a message exchanging among the objects involved, as it is required in an OO environment.

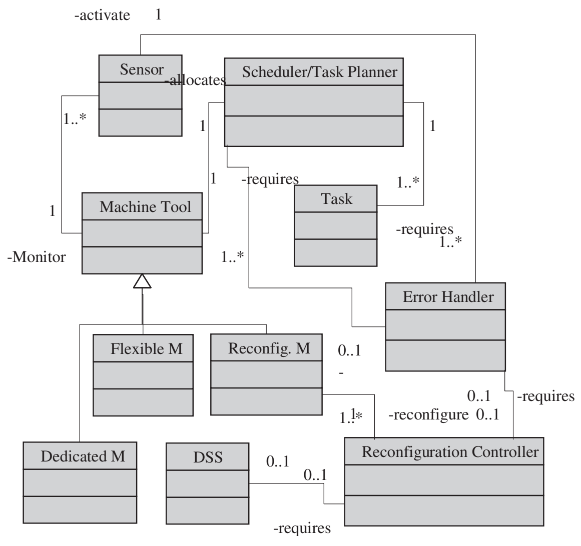

Fig. 4. UML class diagram for the proposed controller.

The sensor advises the scheduler of the occurred error. The scheduler, while stopping the execution of the task plan, requires the error handler for error recovery and this requires the reconfiguration controller to test if the reconfiguration for error handling policy is feasible. Once this last condition has been verified, the error handler asks the DSS to make a decision concerning which error handling policy need to be undertaken and after the DSS feedback, the reconfiguration controller proceeds in reconfiguring the RMT for error recovery.

5. Conclusion

The work presented in this paper investigated operation management issues related to reconfigurable manufacturing systems. Specifically, motivated by the results obtained by using reconfiguration for error handling in scheduling systems, the aim here was to explore the same policy at a cell level. Specifically, it was meant to explore potential of a new operation management approach for handling out-of-ordinary events.

This approach is based on the reconfiguration feature of RMSs. Although reconfiguration cannot be considered as a flexibility feature in terms of ordinary scheduling operations, it can be used instead as a routing flexibility enabling technology

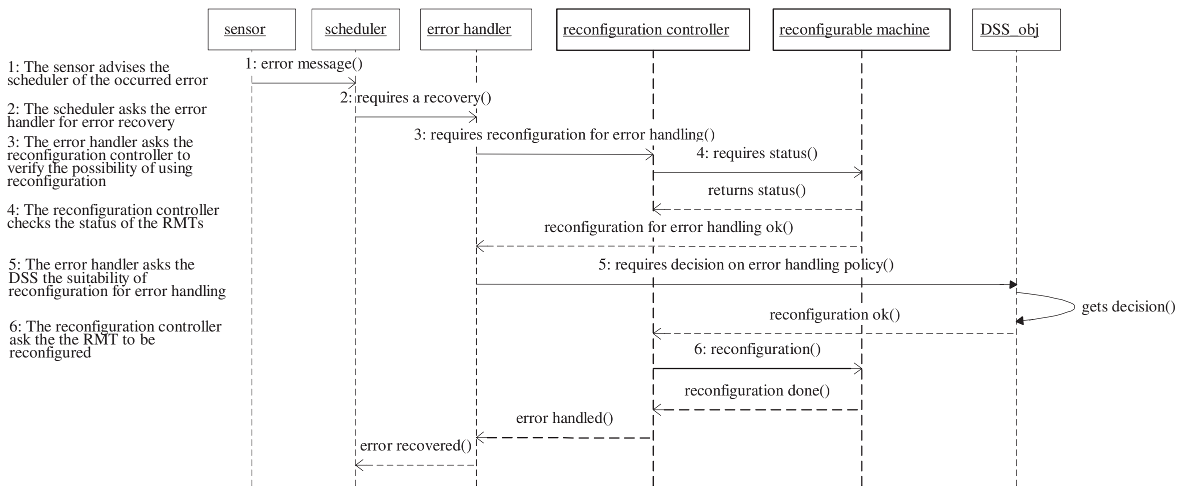
sensor
scheduler
error handler
reconfiguration controller
reconfigurable machine
DSS_obj
1: The sensor advises the scheduler of the occurred error
2: The scheduler asks the error handler for error recovery
3: The error handler asks the reconfiguration controller to verify the possibility of using reconfiguration
4: The reconfiguration controller checks the status of the RMTs
5: The error handler asks the DSS the suitability of reconfiguration for error handling
6: The reconfiguration controller ask the the RMT to be reconfigured
1: error message()
2: requires a recovery()
3: requires reconfiguration for error handling()
4: requires status()
returns status()
reconfiguration for error handling ok()
5: requires decision on error handling policy()
gets decision()
reconfiguration ok()
6: reconfiguration()
reconfiguration done()
error handled()
error recovered()

Fig. 5. UML sequence diagram for the message flow throughout the error handling.

in exceptional cases, as production process errors, such as machine breakdowns, unexpected job arrivals or canceling, tool breakages, robot collisions, sensor damages, and so forth.

Decisions concerning this problem are very complex, as it involves many variables such as the reconfiguration time, the cost associated with reconfiguration, the cost associated with the delay, the time for repairing the machine, the machine availability, the number of tasks in the canceled order, and so forth.

From a deep literature analysis, the problem that surfaced was how to deal with a machine breakdown in a reconfigurable system that does not allow any routing flexibility. In this case, error handling essentially matches with making the decision about if and how a reconfigurable machine should be reconfigured to make the system able to perform those operations that cannot be temporarily performed due to the occurrence of the breakdown of another, associated machine. The complexity of such a decision-making process depends mostly on both performance and economic concerns involved in the problem. An imaginary manufacturing environment has been presented in order to draw attention to the complexity of the above-mentioned decision process. However and mainly, it represents a simple example of how advantages can be achieved by using reconfiguration to handle machine breakdowns. Also, it emphasizes how these advantages are achieved only under particular system constructive characteristics and under specific economic assumptions. The authors proposed a high-level object-oriented control architecture for error handling aided by reconfiguration and presented it by using the unified modeling language notation. It relies on the new idea of using reconfiguration for error handling and integrates it with the existing scheduling or task planning system.

Acknowledgments

Authors are pleased to acknowledge the financial support of the Italian Ministry of University and Research in Science and Technology, which allowed M. Bruccoleri to be a visiting scholar at the NSF ERC for Reconfigurable Machining System at the University of Michigan throughout the presented research.

References

Adacher, L., Agnetis, A., Meloni, C., 2000. Autonomous agents architectures and algorithms in flexible manufacturing systems. IIE Transactions 941–951.

Amico, M., Bruccoleri, M., Noto La Diega, S., Perrone, G., Renna, P., 2001. An intelligent controller for error handling decisions in reconfigurable manufacturing systems. Proceedings of the AITEM Congress, September 2001, Bari, vol. 1, pp. 355–364.

Belz, R., Mertens, P., 1996. Combining knowledge-based systems and simulation to solve rescheduling problems. Decision Support Systems 17, 141–157.

Borchelt, R.D., Thorson, J., 1997. Toward reusable hierarchical cell control software. International Journal of Production Research 35 (2), 577–594.

Brandimarte, P., Rigondanza, M., Roero, L., 2000. Conceptual modeling of an object-oriented scheduling architecture based on the shifting bottleneck procedure. IIE Transactions 921–929.

Bruccoleri, M., Pasek, Z.J., 2002. Operational issues in reconfigurable manufacturing systems: Exception handling. Proceedings of the Fifth biannual World Automation Congress, June 9–13, 2002, Orlando, FL.

Bruccoleri, M., Amico, M., Perrone, G., Pasek, Z., 2002. A distributed control approach for error handling in reconfigurable manufacturing system based on multi agent system technologies. Proceedings of JUSFA Japan-USA Symposium on Flexible Automation, July 2002, Hiroshima, Japan.

Bruccoleri, M., Amico, M., Perrone, G., 2003a. Distributed intelligent control of exceptions in reconfigurable manufacturing systems. International Journal of Production Research 41 (7), 1393–1412.

Bruccoleri, M., Perrone, G., Noto La Diega, S., 2003b. An object oriented approach for flexible manufacturing control system analysis and design using the unified modeling language. International Journal of Flexible Manufacturing Systems 15, 195–216.

Dutta, A., 1990. Reacting to scheduling exceptions in FMS environments. IIE Transactions 22 (4), 300–315.

Gou, L., Luh, P.B., Kyoya, Y., 1998. Holonic manufacturing scheduling: Architecture, cooperation mechanism, and implementation. Computers in Industry 37, 213–231.

Hasegawa, M., Takata, M., Temmyo, T., Matsuka, H., 1990. Modeling of exception handling in manufacturing cell control and its application to PLC programming. Proceedings of 1990 IEEE International Conference on Robotics and Automation. IEEE, Computer Society, Los Alamitos, CA, USA, p. 514–519.

Heng, Li., Li Zhicheng, Li Ling, X., Bin Hu, 2000. A production rescheduling expert simulation system. European Journal of Operational Research 124, 283–293.

Holthaus, O., 1999. Scheduling in job shops with machine breakdowns: An experimental study. Computers & Industrial Engineering 137–162.

Hu, W., Schroeder, M., Starr, A.G., 1999. A knowledge-based real-time diagnostic system for PLC controlled manufacturing systems. Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, vol. 4. IEEE, USA, pp. 499–504.

Jain, A.K., Elmaraghy, H.A., 1997. Production scheduling/rescheduling in flexible manufacturing. International Journal of Production Research 37 (1), 281–309.

Jang, J., Koo, P.H., Nof, S.Y., 1997. Application of design and control tools in a multirobot cell. Computers & Industrial Engineering 32 (1), 89–100.

Jeng, Mu Der, 1997. Petri nets for modeling automated manufacturing systems with error recovery. IEEE Transactions on Robotics and Automation 13 (5), 752–760.

Kao, J.F., 1995. Optimal recovery strategies for manufacturing systems. European Journal of Operational Research 80 (2), 252–263.

Klein, I., Jonsson, P., Backstrom, C., 1996. Automatic synthesis of control programs in polynomial time for an assembly line. Proceedings of the IEEE Conference on Decision and Control 2, 1749–1754.

Kokkinaki, A.I., Valavanis, K.P., 1996. Error specification, monitoring and recovery in computer-integrated manufacturing: An analytic approach. IEE Proceedings: Control Theory and Applications 143 (6), 499–508.

Kopacek, P., Kronreif, G., Probst, R., 1999. A modular control system for flexible robotized manufacturing cells. Robotica 17, 23–32.

Koren, Y., Heisel, U., Jovane, F., Moriwaki, T., Pritschow, G., Ulsoy, G., Van Brussel, H., 1999. Reconfigurable manufacturing systems. Annals of the CIRP 48 (2), Keynote Paper.

Lopes, L.S., Camarinha-Matos, L.M., 1995. Machine learning approach to error detection and recovery in assembly. IEEE International Conference on Intelligent Robots and Systems, vol. 3. IEEE, Piscataway, NJ, USA, pp. 197–203.

Mehrabi, M.G., Ulsoy, A.G., Koren, Y., 2000. Reconfigurable manufacturing systems: Key to future manufacturing. Journal of Intelligent Manufacturing 11, 403–419.

Mehta, S.V., Uzsoy, R.M., 1997. Predictable scheduling of a job shop subject to breakdowns. IEEE Transactions on Robotics and Automations 14 (3), 365–378.

Najjari, H., Steiner, S.J., 1997. Integrated sensor-based control system for a flexible assembly. Mechatronics 7 (3), 231–262.

O'Donovan, R., Uzsoy, R., McKay, K.N., 1999. Predictable scheduling of a single machine with breakdowns and sensitive jobs. International Journal of Production Research 37, 4217–4233.

Park, S.C., Raman, N., Shaw, M.J., 1997. Adaptive scheduling in dynamic flexible manufacturing systems: A dynamic rule selection approach. IEEE Transactions on Robotics and Automations 13 (4), 486–502.

Piramuthu, S., Shaw, M., Fulkerson, B., 2000. Information-based dynamic manufacturing system scheduling. International Journal of Flexible Manufacturing System 219–234.

Sabuncuoglu, I., Bayiz, M., 2000. Analysis of reactive scheduling problems in a job shop environment. European Journal of Operational Research 126, 567–586.

Seifert, R.W., Morito, S., 2001. Cooperative dispatching—exploiting the flexibility of an FMS by means of incremental optimization. European Journal of Operational Research 129, 116–133.

Stevens, P., Pooley, R., 2000. Using UML: Software Engineering with Objects and Components. Pearson Education Limited, Harlow, Essex.

Toguyeni, A.K.A., Craye, E., Gentina, J.C., 1996. Framework to design a distributed diagnosis in FMS. Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, vol. 4. IEEE, Piscataway, NJ, USA, 96CH35929, pp. 2774–2779.

Unal, A.T., Uzsoy, R., Kiran, A.S., 1997. Rescheduling on a single machine with part-type dependent setup times and deadlines. Annals of Operations Research 70, 93–113.

Wu, H.J., 1999. Methodology of generating recovery procedures in a robotic cell. Proceedings—IEEE International Conference on Robotics and Automation 1, 799–804.