Heterarchical control systems for production cells - a case study

Citation for published version (APA):

Mortel-Fronczak, van de, J. M., & Rooda, J. E. (1997). Heterarchical control systems for production cells - a case study. In P. Kopacek (Ed.), *Manufacturing systems : modelling, management and control, MIM'97 : a proceedings volume from the IFAC workshop, Vienna, Austria, 3-5 February 1997* (pp. 213-218). Pergamon.

Document status and date:

Published: 01/01/1997

Document Version:

Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

- A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.

[Link to publication](#)

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

www.tue.nl/taverne

Take down policy

If you believe that this document breaches copyright please contact us at:

openaccess@tue.nl

providing details and we will investigate your claim.

HETERARCHICAL CONTROL SYSTEMS FOR PRODUCTION CELLS — A CASE STUDY

J.M. van de Mortel-Fronczak and J.E. Rooda

*Eindhoven University of Technology, Department of Mechanical Engineering,
P.O. Box 513, 5600 MB Eindhoven, The Netherlands,
E-mail: {vdmortel,rooda}@wtb.tue.nl*

Abstract: Most control systems of flexible production cells have a hierarchical structure. They become very complicated and difficult to maintain and modify when the underlying production cells grow in size and complexity. Moreover, they are characterized by a relatively high sensitivity to failures. As opposed to that, heterarchical control systems are flexible, modular, easy to modify, and — to some extent — fault-tolerant. In this paper, a heterarchical control system of a flexible production cell is formally specified in the CSP-based language χ . This language is well suited for the description of autonomous components cooperating with each other by exchanging information.

Keywords: Modelling, control systems, parallel programs.

1. INTRODUCTION

Most control systems of flexible production cells have a hierarchical structure and are based on the principle of centralized control. In such a structure, different control layers form a pyramid. The lowest layer performs the commands of the layer above it by controlling the production devices. Every layer but the highest one performs the commands of the layer above it by controlling the layer below it. Furthermore, it receives status information from the layer below it and passes (a part of) it to the layer above it. The highest component in the control hierarchy takes care of the realization of system goals.

Hierarchical control systems are rigid and highly sensitive to failures. Moreover, they tend to become very complicated and difficult to maintain and modify when the underlying production cells grow in size and complexity. A change of the configuration of production facilities often requires extensive (and expensive) changes in the associated control system. This explains the growing interest for heterarchical control systems (Lin and Solberg, 1994; Veeramani *et al.*, 1993). Heterarchical control systems are based on the principle

of distributed control and consist of autonomous components cooperating with each other by exchanging information. By means of cooperation, system goals are realized. Heterarchical systems are flexible and modular. They are easy to modify and to adapt in the case of re-configuration or increase in size of production facilities. To some extent, they are also fault-tolerant. However, one should be aware of a possibly high communication overhead inherent to their distributed nature.

The purpose of this paper is to present a formal specification of a heterarchical control system for a flexible production cell. The flexible production cell consists of five machining stations, one quality control station, one input-output station, and an automatic transport system. For the specification, the CSP-based (Hoare, 1985) language χ (van de Mortel-Fronczak and Rooda, 1995; van de Mortel-Fronczak and Rooda, 1996) is used. This language is well suited for the description of autonomous components cooperating with each other by exchanging information. Exchange of information is modelled by synchronous message passing. Component descriptions are concise and comprehensible, and suited for re-use. Specifications defined in χ can be validated by simulation (Naumoski and

Fig. 1. Flexible production cell

Alberts, 1995). Simulation can also be used to assess communication overhead of the specified control system. Additionally, it is possible to formally verify χ specifications (Kleijn, 1996).

The paper is structured as follows. In Section 2, a heterarchical control system structure of a flexible production cell is discussed. The control components are formally described in Section 3. In Section 4, concluding remarks are presented.

2. CONTROL SYSTEM STRUCTURE

In this section, a formal specification is presented of a heterarchical control system of the flexible manufacturing cell schematically drawn in Figure 1. This cell consists of seven workstations (five machining workstations W , one quality-control station Q , and one storage station S) and an automatic transport system. The transport system consists of the conveyor and seven conveyor stations. The conveyor holds a limited number of pallets and moves at a constant speed. Each conveyor station is associated with one workstation and holds up pallets intended for it. Each workstation is equipped with a robot that moves product carriers between the pallet and the workstation. Every pallet carries a magnetic label needed for identification. The production is organized by autonomous job agents. In (Coenen, 1995), a detailed and slightly different model is presented. In this paper, a model is presented on a higher abstraction level.

For the purpose of this paper, it does not matter what kind of workstations belong to the manufacturing system. A relevant assumption is that they work independently. Moreover, for simplicity it is assumed here that storage S can hold an infinite number of parts and products. In practice, storage S also functions as the input/output station. The manufacturing cell can be used to process and/or assemble a family of products (for instance, as described in (van Brussel *et al.*, 1993) or in (Esh, 1994)).

In the model formally specified in the next section:

Fig. 2. Model of the cell and its control

- product carriers are universal, that is, each part or product can be placed on each product carrier,
- assembling is not treated,
- failures are not taken into account,
- by orders, released production orders are meant,
- each order applies to one product.

The structure of the model, as drawn in Figure 2, is based on the structure of the flexible production cell. In the model, the following components are distinguished: order buffer B , job agents J , communication network N , workstation agents A , transport system T , and workstations W , Q , and S . Except for the storage, the workstations can be modelled in the same way on this abstraction level. In general, several storage components are allowed.

The components B , J , N , and A belong to the control system of the flexible production cell. The job agents take care of the realization of orders that arrive and leave the cell through buffer B . Every order specifies a product to be produced by defining the raw material type (part) and the sequence of operations leading from this raw material to the product in question. The first operation of each order is a retrieve operation for a storage component.

To realize an order, every job agent repeats the following sequence of steps for every operation defined in the order. It inquires, through the communication network, which workstation is willing to perform the operation. If a few workstations respond to the inquiry, one of them is chosen accord-

Fig. 3. Subsystem N

ing to a certain criterion. (In the model presented in the next section, the operation is assigned to the first workstation that responds to this inquiry.) The workstation chosen has to wait for the arrival of the pallet carrying the part that has to be transformed. The job agent informs the storage (if the operation is the first one of the sequence) or the workstation that performed the previous operation about the destination of the associated part, so that they can take care of passing the part (with the proper destination address) to the transport system. The transport system then delivers the part to its destination. When the workstation has completed its task, it informs the job agent about this fact and the steps described above can be repeated. The last operation specified in an order is the operation of putting the finished product in the storage.

Subsystem N consists of a number of interface components I collecting messages for and from the components connected to the communication network and switch S that takes care of passing the messages from a source to a destination. The structure of subsystem N is graphically presented in Figure 3.

The components in the model cooperate with each other by exchanging information via communication channels (arrows in the figures) according to a communication protocol.

In the sequel, only components B , J , N , and A associated with the cell control system are specified in detail. For models of workstations and of the transport system the reader is referred to (Coenen, 1995).

In the specification language χ , the following notations are used for dealing with data and defining data types:

- T^* denotes the collection of lists with elements of type T (for instance, nat^* — lists of natural numbers),
- $[1, 2, 3]$ denotes the list of numbers 1, 2, and 3,
- $[]$ denotes the empty list,
- $xs \uparrow ys$ denotes concatenation of lists xs and

ys ,

- $\text{hd}(xs)$ denotes the first element of xs (the head),
- $\text{tl}(xs)$ denotes the list that results from removing the first element of xs (the tail),
- $T_1 \times T_2 \times \dots \times T_n$ denotes the collection of n -tuples where the i th element is of type T_i (Cartesian product; for instance, $\text{nat} \times \text{real}$ — pairs of natural and real numbers),
- the elements of n -tuple p are referred to as $p.i$, for $0 \leq i < n$,
- $\langle 1, c, 7, 30, 5 \rangle$ denotes a tuple of five elements: 1, c , 7, 30, and 5.

To model material and information flows, the following data types are used :

- $\text{ptype} = \{1, 2, \dots, n\}$ — numbers corresponding to n part and product types,
- $\text{op} = \{1, 2, \dots, m\}$ — numbers corresponding to m operations that can be performed in the production cell,
- $\text{task} = \text{ptype} \times \text{op}$ — combinations of parts and operations to be applied to them,
- $\text{order} = \text{ptype} \times \text{ptype} \times \text{task}^*$ — orders with the following contents: a product number, a part number, and a sequence of tasks that transforms the part into the specified product,
- $\text{wid} = \{0, 1, \dots, 6\}$ — workstation identification numbers (7 workstations),
- $\text{jid} = \{7, 8, \dots, j+6\}$ — job agent identification numbers (j job agents),
- $\text{id} = \text{wid} \cup \text{jid}$ — identification numbers of both the workstations and the job agents,
- $\text{info} = \text{id} \times \text{id}^* \times \text{char} \times \text{nat} \times \text{task} \times \text{wid}$ — network messages with the following contents: source, message destination list, kind of message, message number, task number, destination for a (semi-finished) product (only for d-messages as explained in the next section),
- $\text{cinfo} = \text{id} \times \text{char} \times \text{nat} \times \text{task} \times \text{wid}$ — information for workstation controllers: as described above except for the message destination list.

In this paper, only information flow is treated.

3. DESCRIPTION OF CONTROL COMPONENTS

The components in the model are called processes in the specification language χ . In the heading of each process the communication channels and, possibly, parameters are defined. For instance:

- $pb : ? \text{order}$ in process B defines input channel pb over which orders are received,
- $bp : ! \text{ptype}$ in process B defines output channel bp over which information about finished

products is sent,

- $is : ?(\text{info}^{j+7})$ in process S defines a bundle of $j+7$ input channels $is.0$ through $is.(j+6)$ over which network messages are received,
- $si : !(\text{info}^{j+7})$ in process S defines a bundle of $j+7$ output channels $si.0$ through $si.(j+6)$ over which network messages are sent,
- $id : \text{jid}$ in process J defines a parameter id of type jid .

The body of each process consists of variable declarations followed by executable statements modelling a components behaviour. The executable part begins with some initial communication actions and/or variable initialization. Furthermore, it usually contains an infinite repetition in which after a (conditional) communication action some other statements are performed. The enabled communication actions are executed in the first-in-first-out fashion.

Order buffer B receives orders ($pb?o$) and stores them in the list os ($os := os \uparrow [o]$). Orders are distributed among the free job agents ($ba.i!hd(os)$) and subsequently removed from the list os ($os := tl(os)$). When an order is completed the associated job agent sends the information about the finished product type back to the order buffer ($ab.i?p$). The information about finished products is stored in the list ps ($ps := ps \uparrow [p]$). On request, this information can be passed through ($bp!hd(ps)$) and subsequently removed from the list ps ($ps := tl(ps)$).

```

proc B( pb : ? order, bp : ! ptype, ba : ! (orderj)
      , ab : ? (ptypej)
      ) =
  [ o : order, os : order*, p : ptype, ps : ptype*
  | os := []; ps := []
  ; * [
      pb?o      → os := os ↑ [o]
      [ i : len(os) > 0; ba.i!hd(os) → os := tl(os)
      [ i :
          ab.i?p      → ps := ps ↑ [p]
          [ len(ps) > 0 ; bp!hd(ps) → ps := tl(ps)
          ]
      ]
  ]

```

Job agent J takes care of the completion of each order received from the order buffer ($ba?o$). To this end, for each task in the list it inquires the workstation agents about the possibilities to perform the task ($an!(id, ds, q, nr, hd(ts), 0)$). The first workstation agent that responds to this inquiry (which message p satisfies: $p.2 = r \wedge p.3 = nr$) is charged with the task ($an!(id, [nw], c, 0, hd(ts), 0)$). The strategy of assigning tasks to responding workstations differs from the one described in (Coenen, 1995). There, the workstation that responded in a determined time slot and that has the smallest amount of work to do is charged with the task. This results in a slightly more complicated model but does not contribute to the general idea of the model discussed

in this paper. For each task of the list but the first one, the job agent also has to inform the workstation that performed the previous task about the new destination of the semi-finished product ($an!(id, [ow], d, 0, 0, nw)$). This is achieved by using variable f that changes from true to false after the first task in the list is completed. Each time a task is completed, the job agent receives message p containing this information ($p.2 = s$). When all tasks in the list have been performed, the information about the finished product type is sent to the buffer B ($ab!o.0$).

```

proc J( ba : ? order, ab : ! ptype, an : ! info
      , na : ? info, id : jid
      ) =
  [ nr : nat, nw, ow : wid, o : order, ts : task*
  , ds : wid*, b, f : bool, p : info
  | nr := 0; nw := 0; ds := [0, 1, 2, 3, 4, 5, 6]
  ; * [ ba?o
      → ts := o.2
      ; f := true
      ; * [ len(ts) > 0
          → nr := nr + 1
          ; an!(id, ds, q, nr, hd(ts), 0)
          ; b := false
          ; * [ ¬b; na?p
              → b := p.2 = r ∧ p.3 = nr
              ]
          ; ow := nw
          ; nw := p.0
          ; an!(id, [nw], c, 0, hd(ts), 0)
          ; [ f → f := false
              [ ¬f → an!(id, [ow], d, 0, 0, nw)
              ]
          ; b := false
          ; * [ ¬b; na?p → b := p.2 = s ]
          ; ts := tl(ts)
          ]
      ]
  ; ab!o.0
  ]

```

To model the network, two types of components are defined: interface I and switch S . Interface I is just a buffer for messages sent by or destined to job agents and workstation agents. In general, it can be said that it is a message buffer for components connected to the network.

```

proc I(an, si : ? info, na, is : ! info) =
  [ p : info, pi, po : info*
  | pi := []; po := []
  ; * [
      an?p      → pi := pi ↑ [p]
      [ len(pi) > 0; is!hd(pi) → pi := tl(pi)
      [
          si?p      → po := po ↑ [p]
          [ len(po) > 0; na!hd(po) → po := tl(po)
          ]
      ]
  ]

```

The task of switch S is to deliver the messages sent by job and workstation agents to their desti-

nations. A message can be received from any interface ($is.i?p$) connected to the switch. The destination list included in the message is stored in ds ($ds := p.1$). Subsequently, the message is sent to all interfaces associated with the destination list ($*[\text{len}(ds) > 0 \rightarrow si.\text{hd}(ds)!p; ds := \text{tl}(ds)]$).

```

proc S(is : ?(infoj+7), si : !(infoj+7)) =
  || p : info, ds : wid*
  | *[ i : is.i?p → skip ]
    ; ds := p.1
    ; *[ len(ds) > 0 → si.hd(ds)!p; ds := tl(ds) ]
  ||

```

Workstation agent A communicates with job agents. It can receive ($na?p$) one of three kinds of messages: a request ($p.2 = q$), a confirmation ($p.2 = c$), or a destination ($p.2 = d$). It passes the message received (except for the destination list) to the workstation controller ($ac!(p.0, p.2, p.3, p.4, p.5)$). The workstation controller uses different parts of the message depending on its kind (q : task number; c : source, message number, task number; d : destination for the (semi-finished) product). In the case of a request, the workstation controller gives in response the information whether or not the workstation is able to perform the specified task ($qa?b$). If true (b) the workstation agent sends a positive replay to the requesting job agent ($an!(id, [p.0], r, p.3, 0, 0)$). When the task assigned to the workstation is completed, the workstation controller informs the workstation agent about it ($ca?c$). The workstation agent passes this information to the proper job agent ($an!(id, [c.0], s, c.1, 0, 0)$).

```

proc A(na : ?info, an : !info, ac : !cinfo,
      qa : ?bool, ca : ?(id × nat), id : wid) =
  || p : info, b : bool, c : (id × nat)
  | *[ na?p
    → ac!(p.0, p.2, p.3, p.4, p.5)
    ; [ p.2 = q
      → qa?b
      ; [ b → an!(id, [p.0], r, p.3, 0, 0)
        || ¬b → skip
      ]
    || p.2 ≠ q
    → skip
    ]
  || ca?c
    → an!(id, [c.0], s, c.1, 0, 0)
  ||

```

Components can be grouped in systems. In the heading of each system the communication channels and, possibly, parameters are defined. The body of each system consists of internal channel declarations followed by the enumeration of the components (processes or subsystems) with ap-

propriate channel names and parameters. Internal channels connect pairs of system components and are therefore always listed in two instances. This way interaction path in the system are established. The components in a system definition are separated by $\parallel$ symbol which stands for parallel composition. In the case of channel bundles and various instances of one process definition an abbreviation may be used as shown below.

Formally, subsystem N graphically presented in Figure 3 is defined by:

```

syst N(an : ?(infoj+7), na : !(infoj+7)) =
  || is : (infoj+7), si : (infoj+7)
  | S(is, si)
  || i : 0 ≤ i < j + 7 : I(an.i, si.i, na.i, is.i)
  ||

```

Specification of the remaining components (from Figure 2) is beyond the scope of this paper.

4. CONCLUDING REMARKS

In this paper, a specification of a heterarchical control system for a flexible production cell is presented. In particular, the part that takes care of the organization of production is specified in detail. Descriptions of the components are concise and easy to change if a different strategy of assigning tasks to workstations is to be applied. The number of job agents is a parameter of the system, so it can easily be varied. However, in a particular system instance, this number is fixed. The control system is fault-tolerant to some extent: a workstation failure does not influence the functioning of the remaining parts of the production cell. It is also modular and easy to modify if the configuration of production facilities would change.

The specification language used has a parallel nature and is therefore well suited for the description of heterarchical control systems. Modelling with χ is supported by a software system including a simulator for systems defined in this language.

REFERENCES

- Coenen, F.W.J. (1995). Een heterarchische besturing voor flexibele productiesystemen. Master's thesis. Eindhoven University of Technology, Department of Mathematics and Computing Science. Eindhoven, The Netherlands. (In Dutch).
- Esh (1994). *Computer Integrated Manufacturing for Industrial Training Applications*.
- Hoare, C.A.R. (1985). *Communicating Sequential Processes*. Prentice-Hall International. London, UK.
- Kleijn, J.J.T. (1996). Verifying a χ specification with the aid of PSF and ACP. Report WPA

420104. Eindhoven University of Technology, Department of Mechanical Engineering. Eindhoven, The Netherlands.
- Lin, G.Y. and J.J. Solberg (1994). Autonomous control for open manufacturing systems. In: *Computer Control of Flexible Manufacturing Systems: Research and Development* (S.B. Joshi and J.S. Smith, Eds.). pp. 169-206. Chapman and Hall. London.
- Naumoski, G. and W.T.M. Alberts (1995). The χ engine: a fast simulator for systems engineering. Final report. Eindhoven University of Technology, Stan Ackermans Institute. Eindhoven, The Netherlands.
- van Brussel, H., Y. Peng and P. Valckenaers (1993). Modelling flexible manufacturing systems based on Petri nets. *Annals of the CIRP* 42(1), 479 - 484.
- van de Mortel-Fronczak, J.M. and J.E. Rooda (1995). Application of concurrent programming to specification of industrial systems. In: *Proceedings of INCOM'95*. Beijing, China. pp. 421-426.
- van de Mortel-Fronczak, J.M. and J.E. Rooda (1996). On the integral modelling of control and production management systems. In: *Proceedings of APMS'96* (N. Okino, H. Tamura and S. Fujii, Eds.). Kyoto, Japan. pp. 171-176.
- Veeramani, D., B. Bhargava and M.M. Barash (1993). Information system architecture for heterarchical control of large FMSs. *Computer Integrated Manufacturing Systems* 6(2), 76-91.