5

An Interaction Model for Reconfigurable Process Control

5.1 Introduction

The previous chapter developed a reconfigurable architecture for process control comprising four basic types of process elements. As discussed these elements must interact in order to exchange information and make collective decisions required for a complete plant-wide control. In this chapter we develop a model to support such interactions so as to describe how elements carry out information exchanges in order to produce the required end-products. The proposed model is a direct and necessary supplement to the reconfigurable control architecture developed in the previous chapter.

5.1.1 Overview

We again start by defining the term *interaction* and the meaning of an *interaction model* in the context of a distributed control architecture.

The term interaction – defined in a dictionary term as *action or influence of persons or things on each other* (Oxford English Dictionary, 2005) – has different meanings when referred to distributed architectures.These include: (i) *communication* (a pre-determined, passive message-passing protocol), (ii) *collaboration* (communication with dynamic selection of messages from an application-specific library), and (iii) *coordination* (collaboration with an ability to reason about and synthesise messages dynamically) (Wooldridge 2002). While the first two forms are used vividly in day-to-day life, it is the third form that has found convincing use in distributed decision-making applications such as in multi-agent systems, manufacturing control, supply chain management, *etc.* Coordination is also the meaning used in our interaction model, however, the current chapter only focusses on the collaboration aspects as defining the structure of information exchanged between process elements. The strategies for reasoning about local decisions are addressed in the next chapter.

The previous work in holonic or agent-based research has generally used contracting or its extended variant as the means for defining the message

structure for inter-element interactions between product and resource holons. As discussed in Section 3.2, contracting alone cannot be applied to continuous processes because of the various restrictions such as the tight and finite interconnections between process units and the continuous flow of materials. These constraints can make the interactions significantly complex if the responsibility of managing unit, header or service elements is passed solely to product elements. Instead, we seek an interaction approach that allows these three latter elements to coordinate their operations directly among themselves while also interacting with product elements.

We seek inspiration from the research on information and life-cycle management in supply chain and virtual enterprise fields (Camarinha-Matos *et al.* 2003, Strader *et al.* 1998). The proposed model is then framed around developing an interaction method that process elements can use to implement the entire reconfiguration process shown earlier in Fig. 3.1 in a distributed manner, *i.e.,* how they identify the need for reconfiguration; define the new configuration; reorganise the process schemes; and, deliver the order requirements. The interaction model essentially builds upon a key concept analogous to market transactions between companies: a customer process element (*e.g.*, a unit element) that needs to acquire a material or service for its task can buy it from a supplier process element that can supply it, *i.e.,* the interactions between unit, header and service elements in the DRPC architecture are modelled as forms of supplier-customer type transactions between companies in a supply chain.

5.1.2 Requirements for RPC Interaction Model

The interaction model (including the coordination strategy to be developed in the next chapter) is aimed to deliver the dynamical aspects of reconfigurability requirements, in particular, product and process diversity, responsiveness and fault-tolerance. In addition, it should retain the modularity of the distributed architecture, *i.e.,* it should not impose any centralised information constraints that can disturb the modifiability of the RPC system. The key focus of this chapter is then to define the method for initial integration and its subsequent refinement of product recipe information with production capabilities in the plant so as to address these requirements in a distributed manner.

This chapter is structured as follows. The next section specifies the structure of process elements' interactions to implement the process of reconfiguration. The key important phases of this model are also explored with details about how the low-level message-passing protocols are organised for exchange of materials and services between elements. Section 5.3 then explains the interaction model by applying it to a small process example. Section 5.4 finally discusses the key features of the model by comparing it against the above requirements and the previous conventional and distributed approaches.

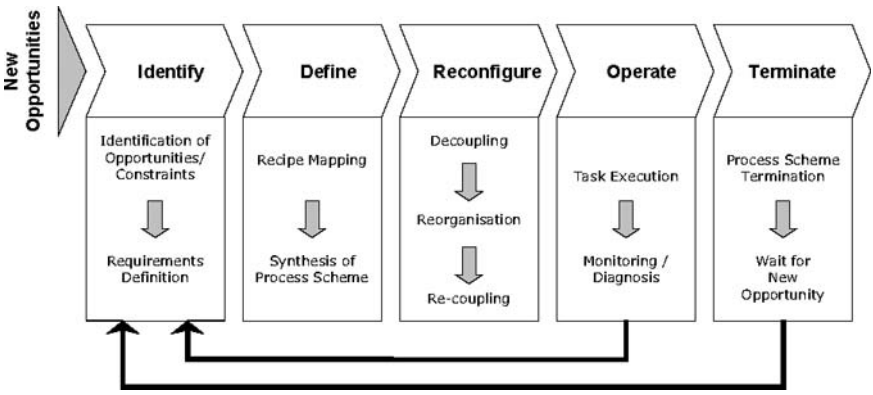

Fig. 5.1. Distributed reconfiguration process

5.2 Specification of the Interactions Between Process Elements

Fig. 5.1 depicts the distributed reconfiguration process to define the interactions of process elements in the DRPC architecture. The figure is developed from an earlier Fig. 3.1, with an additional 'Terminate' phase is now included and the overall activities of reconfiguration are split into five key phases.

We now describe each of these phases in a brief detail. To provide consistency in the discussions, we use the terms *product recipe*, *processing task*, and *process scheme* in the description below. A *product recipe* in the sense of ISA-S88 standard (ANSI/ISA 1995) refers to the minimum set of information that uniquely defines the manufacturing requirements for a specific product, *i.e.,* it identifies raw-materials, their relative quantities and the required processing, but without referring to particular equipment. A *processing task* in a product recipe refers to a unit operation (*e.g.*, reaction) that converts its incoming feedstocks to outgoing products. The recipe may define relative quantities of materials (and services if used) but not the exact values as they depend on the throughput of final products. A product recipe is then defined as a sequence of processing tasks that converts the main raw-materials to final products. A *process scheme* refers to the physical and control configuration of a segment of process that meets the requirements of product recipe for a specific order. Multiple process schemes may exist in the same process for different product orders sharing some unit, header or service elements common between them. Not all unit, header or service elements may be part of a process scheme though, *i.e.,* they can be idle at times and await for a new product order to arrive that is relevant to them.

- **Identify Phase:** The reconfiguration starts with identifying an opportunity for production (when a new production order arrive) or deciding

whether to adapt the ongoing process schemes (when a change occurs). Such changes in particular can be planned or unplanned and can provide with a new opportunity (*e.g.*, availability of a unit element, raw-material or service) or impose a constraint (*e.g.*, a unit element fails or becomes bottleneck). Subsequently, each of the opportunities or constraints are refined into the detailed requirements for reconfiguration of the process. In case of a new customer order, an appropriate new product element is created. (Note that these requirements are not defined explicitly anywhere or centrally within an element; they only define the goal with which the process elements set out to initiate a new round of interactions, for example, to develop a new process scheme when a new customer order arrives.)

- **Define Phase:** The define phase is divided into two sub-phases: (i) *recipe mapping*, to map the product recipe information onto production capabilities available in the plant, and (ii) *synthesis*, to derive a specific process scheme from the potential choices created by recipe mapping.

  - *Recipe Mapping:* In the recipe mapping phase, the product elements associated with customer orders interact with the unit and header elements in the plant to assign (or refine already assigned) processing tasks in product recipes with the production capabilities available in the plant. The interactions lead to a number of tentative process schemes which could be used in the production. Not all tentative schemes may be feasible though because the selection of specific unit or header elements or their operational settings are not defined yet.

  - *Synthesis:* Next, in the synthesis phase, the unit elements involved in the tentative schemes interact among themselves as well as with the header and service elements to refine these tentative schemes into a single scheme that can be implemented for the production. The elements use a global production goal, such as production cost, to arrive at the solution. It is possible that some elements may be already engaged with other process schemes. Such elements interact among associated elements in those schemes to identify how should they reconfigure their operations. Eventually, all concerned elements agree on the structure of the new process scheme including various network parameters (process routes, material and service flow rates, *etc.*) and the operating settings of participating elements (reaction temperatures, heat exchanger duty *etc.*).

    As explained later in this section, the unit elements build the process scheme in an incremental, bottom-up manner by following demand-pull type interactions. In doing so, they also solve both a scheduling problem (defining the configuration of the scheme, *i.e.*, exact assignment of tasks to equipment) and an optimisation problem (defining the config-

uration of control structures and associated local settings).

- **Reconfigure Phase:** The procedure for reconfiguring the process scheme can now begin. The activities can be split conceptually into three sub-phases: decouple, reorganise and recouple, where any or all three of them may involve a physical and/or a control change. A good example of physical change can be a change in the process routes. The header elements involved in this change switch the routes from their current configuration to agreed target configuration. A systematic operating procedure may be required to meet the physical process constraints such as mixing hazards, cleaning-in-place, *etc.*

- **Operate Phase:** As the process scheme is being established, the flow of materials and services can also begin. The unit elements along the route start executing their processing tasks in a coordinated manner so as to convert the incoming feedstocks to their products. The coordination of all activities leading up to the start-up and subsequent on-load operations is achieved by unit elements themselves. The aim is to maintain the process at agreed set-points during synthesis phase.

  During continued operations, the plant conditions may change, *e.g.*, a unit element fails or the customer demand changes. The process elements affected by the change respond to it in a graceful manner. The elements situated next to the point of change attempt to absorb it to the level possible within their local capacity. If this is not achievable, the residual change is propagated further in the process up to a point where it can be fully absorbed. The elements affected along the route adapt their operations as appropriate. If the change or disturbance is small in magnitude or is not likely to last long, then the elements may prefer to operate in this mode for a required period. Only if it is large in magnitude or if the resulting performance is not acceptable, the elements should re-enter into a new round of interactions to reconfigure the process scheme starting with the identify phase.

- **Terminate Phase:** The process scheme is finally dissolved once the throughput requirements for the order are met or if a major failure occurs (such as a reactor element fails) which requires terminating the order altogether. In either case, the process elements involved in the scheme either join other process schemes or idle themselves and wait for a future order to arrive.

In what follows, we elaborate on the two sub-phases – recipe mapping and synthesis – of the define phase, as they are the most critical in terms of how the elements define the structure of a new process scheme. A specification of the underlying interactions described below would depend on the final application and is not developed in great detail here.

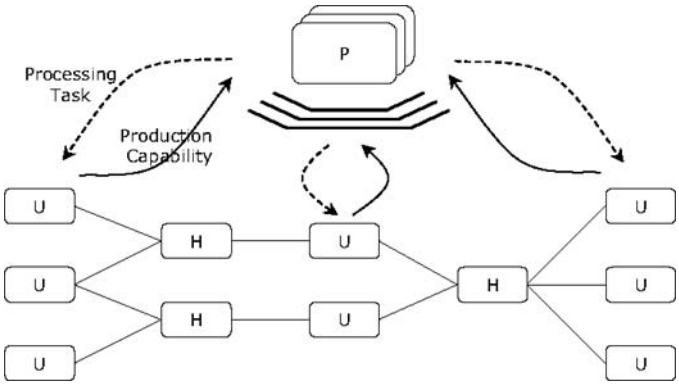

Fig. 5.2. Interactions between product and unit elements during the recipe mapping

5.2.1 Recipe Mapping Phase

Fig. 5.2 illustrates the structure of the interactions during recipe mapping phase. To illustrate the different possibilities, we consider two alternative ways in which this mapping can be carried out: (i) product-centric approach and (ii) unit-centric approach.

Product-Centric Approach

In a product-centric approach, the product elements take the leading role in assigning the processing tasks to unit elements. Each product element announces each of its tasks in the product recipe to all unit elements existing in the plant. The unit elements capable of undertaking the tasks reply back. The replies may contain primary information about the nature of unit operation, *e.g.*, 'reaction type 1', that the unit elements can perform. Based on the replies, the product element identifies the unit element(s) that best suite the currently announced task and assigns the task to them. It also assigns appropriate materials and services as well as other processing requirements for that task (*e.g.*, product quality) to selected unit elements. The interactions repeat in this manner until all tasks in the product recipe are assigned to appropriate unit elements. Note that the same task may be assigned to more than one unit elements but not all may be able to undertake the task, because the information on the connectivity of these unit elements or their local settings is not defined as yet. The interactions thus result in one or more tentative process schemes that are refined into a single scheme in the synthesis phase.

Unit-Centric Approach

The centralised role of product elements in a product-centric mechanism can become bottleneck if they have to match a large number of tasks or the same

tasks more frequently. This latter scenario can arise if the process is required to produce the same end-products more frequently and/or in a highly reconfigurable manner (*e.g.*, in case of many polymer plants).

Instead of product elements, the responsibility of recipe mapping can be distributed among unit elements themselves. The unit elements can now be defined with additional details about the specific processing tasks they can perform and the materials and services they need to acquire to execute these tasks. For instance, a distillation column can be specified with two specific distillation tasks $X \Rightarrow \{Y, Z\}$ and $L \Rightarrow \{M, N\}$ (where X, Y, Z, L, M and N are the materials). The product elements can still be supplied with some form of product recipe or parts of it if the customer order requires only certain processing tasks to be used in making the product.

Using recipe-specific details, the unit elements can be asked to identify which task(s) they can use to produce a specific material. The recipe mapping activity then proceeds as a backward search starting from the unit elements that can produce the final product. These unit elements first identify the tasks they can use to produce the product and the incoming materials they require from other unit elements in the upstream. The unit elements may check with product elements whether their selected tasks are not restricted in the recipe (if supplied). The interactions repeat from the upstream unit elements until the unit elements requiring the main raw-materials are reached. The interactions thus result in one or more tentative process schemes which could be refined into a single scheme during the synthesis phase. Since the synthesis phase is also carried out by unit elements themselves (together with header and service elements) it is possible that the recipe mapping and synthesis can proceed together.

Discussion

It can be seen that both approaches to recipe mapping have their benefits and disadvantages – the product-centric approach may require less time to set up initially while the unit-centric approach may provide increased freedom to unit elements to choose or alter their tasks. The unit-centric approach requires unit elements to be defined with additional recipe information on processing tasks they can perform. This is not a requirement for product-centric case. The initial effort required to set up a unit-centric approach may thus be higher. However, the unit-centric approach also allows unit elements to select or alter their processing tasks that best match with the changing plant conditions. This is instead of those specified by the product elements in a product-centric approach. The unit-centric approach should thus be able to better utilise the plant facilities than the product-centric approach.

We must note that the inclusion of recipe-specific information in unit elements in the unit-centric approach does not change or violate the basic assumption that this product information should be kept separate from production capabilities as considered in ISA-S88 standard (ANSI/ISA 1995) or

the previous distributed research. What the approach suggests is to derive this information in a bottom-up manner by collecting together the tasks of unit elements via their direct interactions into a single scheme. As mentioned earlier in this section, an approach of this nature may be useful when the same products are produced more frequently or the same tasks are reused in different products.

5.2.2 Synthesis Phase

Having identified the processing tasks, the unit elements in the tentative process schemes interact among themselves and with respective header and service elements to identify the structure of a specific process scheme that can be used for production. The interactions are considered to follow a backward-search pattern based on demand-pull in which the unit elements, starting from the terminal stage of the process, attempt to incrementally allocate their material demands to unit elements situated upstream and also the service demands to appropriate service elements. Fig. 5.2 outlines the nature of these interactions.

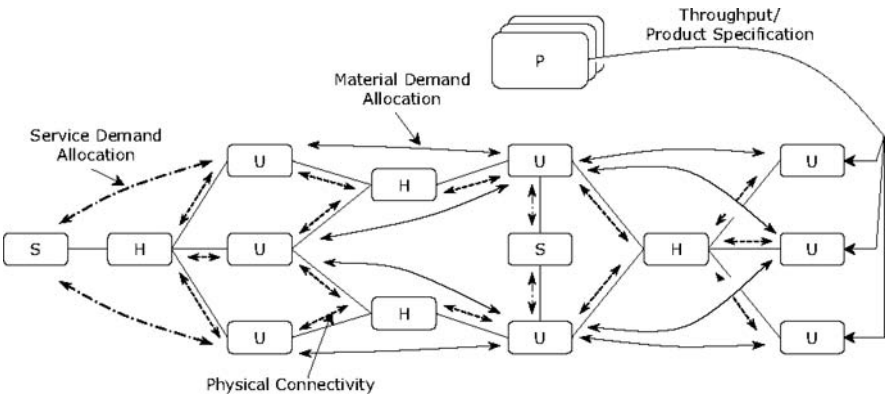

Fig. 5.3. Interactions between unit, header and service elements during the synthesis phase

Internal Design of Process Elements

To model the interactions in detail, we borrow an analogy from the concept of so-called *transaction* between companies in a market or a supply chain in that 'a process element (which can be a unit or a header element) that requires a material or service for execution of its processing task can *buy* this from any other process element or elements which can supply it', *i.e.,* the exchange of

a material or service between process elements can be modelled as a form of contract between two or more different parties in a market or supply chain.

Using this analogy, we can impose a structure on the internal design of process elements, in particular on their coordination modules in Fig. 4.4. A process element which requires access to a material or service for its task can be represented as a customer and a process element that supplies a material or service can be represented as a supplier. Fig. 5.4 depicts this structure. This suggests that each unit element can be modelled as: (a) the supplier of its outgoing products and services (*e.g.*, heat released from exothermic reaction) and (b) the customer of its incoming feedstocks and services. Similarly, each header element can be modelled as a supplier or customer for its supply or use of services and each service element purely as the supplier of its services.

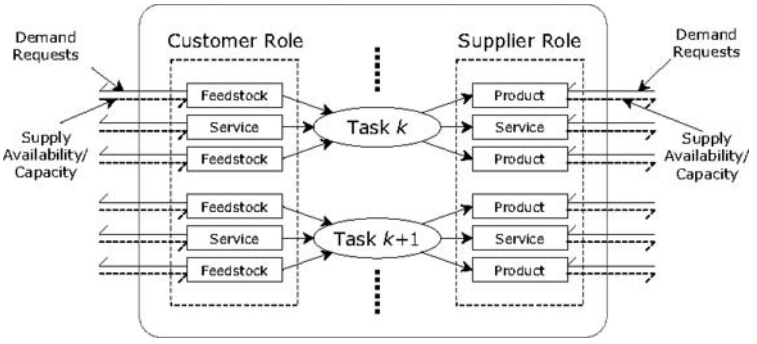

Fig. 5.4. Internal design of process elements based on supplier/customer roles

Interaction Protocol

Based on the supplier-customer roles, the interaction protocols for allocation of material or service demands between two or more process elements can be defined to follow a specific time line:

- Step 1: the customer element announces a demand request for supply of relevant material or service;
- Step 2: the supplier elements which can supply the material or service respond to these requests;
- Step 3: the customer and supplier elements agree on the allocation of material or service demand in terms of respective process parameters, *e.g.*, process flow rates, temperature, pressure.

If the exchange of material or service is to occur via a header element, then those header elements also get involved in the interactions so as to agree on the process routes though which the transfer should occur as well as the

requirements for transforming the physical state of the material or service being transferred, *e.g.*, heat or cool them.

Figs. 5.5 and 5.6 show the interaction protocols for material and service demand allocation between a customer process element (denoted as theCustomer) and one or more supplier process elements (denoted as theSupplier). The exchange occurs via intermediate header elements (denoted as theHeader). In the case of material exchange, the customer element must be a unit element, while the supplier element can be other unit element or an external supplier when it is the main raw-material. Similarly, for the service exchange, the customer element can be a unit or header element while the supplier can be a unit, header or service element depending on where and how the service is supplied.

Note that the protocols in Figs. 5.5 and 5.6 differ in the way the interactions between elements are organised. In a material exchange, the customer elements initiate the interactions for distributing material demands among possible supplier elements; the customer elements therefore act as the coordinators of demand allocations. In a service exchange, again the customer elements initiate the interactions, however the coordination of interactions in terms of the distribution of service is achieved by supplier elements, *i.e.*, the supplier elements act as the coordinators. The computational methods for implementing these protocols therefore must differ.

Synthesis of a Complete Process Scheme

The synthesis of a complete process scheme from tentative process schemes identified during recipe mapping occurs via a sequence of nested material and service exchanges between unit, header and service elements. Fig. 5.7 on page 83 depicts an overview of these interactions between unit elements. All unit elements therein are shown both as customers and suppliers of their feedstocks and products as well as services.

The round of interactions starts from unit elements in the last stage. These unit elements initiate the protocol for material and service allocation in Figs. 5.5 and 5.6 by announcing the demand requests for their feedstocks and services. For material demands, the interactions proceed in backward direction. Not all unit elements in the upstream in tentative schemes which can supply feedstocks may respond because there may exist constraints such as limited connectivity or physical capacity limits; the connectivity information is supplied by the header elements. The unit elements which can meet the supply requirements further initiate a new set of material and service allocation protocols to source their feedstocks from unit elements in further upstream and services from appropriate service elements. The interactions thus repeat until unit elements in the first stage of the process are reached that can acquire their feedstocks from raw-material suppliers. At this stage, starting from the first stage, all concerned unit elements return back with their supply proposals (including the availability and capacity details) to respective customer

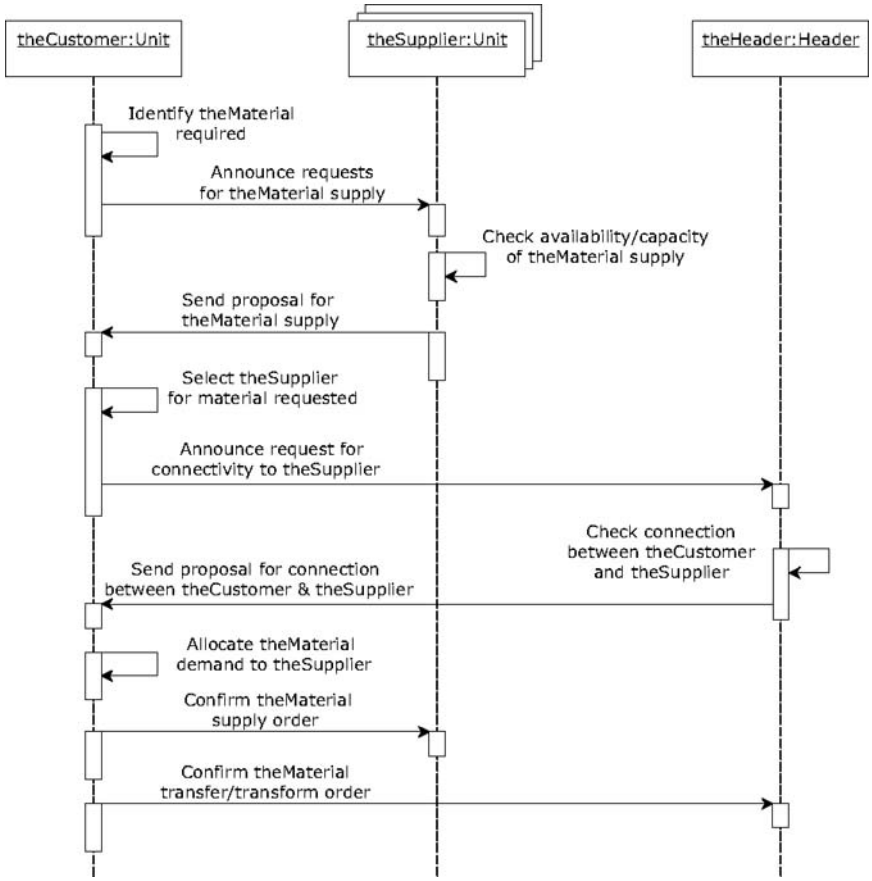

Fig. 5.5. Interaction protocol for material demand allocation

elements. The supply proposals thus flow in forward direction towards the terminal stage. From the responses, each unit element selects which supplier elements are appropriate, and how much material demand it should allocate to them. If necessary, this nested sequence of interactions for material and service allocations repeats until all participating elements settle on respective parameters for material and service demands as well as the process routes through which the transfers should occur. The process scheme thus developed is then reconfigured in the next 'reconfigure' phase which is not described here in detail.

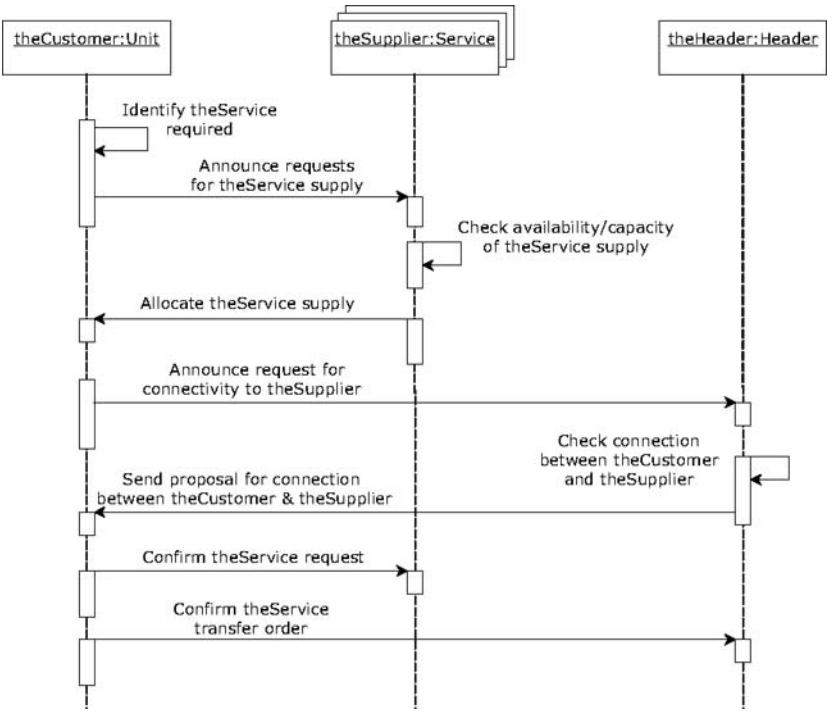

Fig. 5.6. Interaction protocol for service demand allocation

5.3 An Illustrative Example

We next consider a simple process example to illustrate the nature interactions between process elements in the DRPC interaction model. The example illustrates a production start-up scenario where a specified product is to be produced at a given throughput rate. A further detailed example depicting the general production control scenarios such as multiple products is considered in Chapter 7.

5.3.1 Process Description

We consider a process where a product A is produced using the product recipe shown in Fig. 5.8. Each rectangle in the figure represents a material and each oblong a processing task. Each task is associated with at least one outgoing and one incoming material, whereas each material with at least one task. The recipe is of a *non-linear* nature, *i.e.*, there exist two different tasks T3 and T4 that both can produce material D and hence be involved in producing A. Fig. 5.9 depicts the layout of the physical process comprising a set of unit, header and service elements.

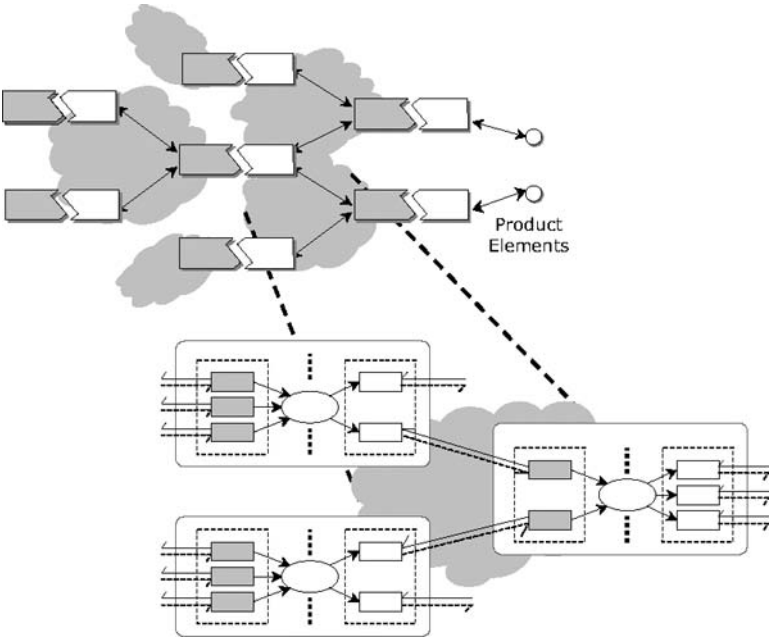

Fig. 5.7. Interactions between unit elements in the synthesis of a complete process scheme

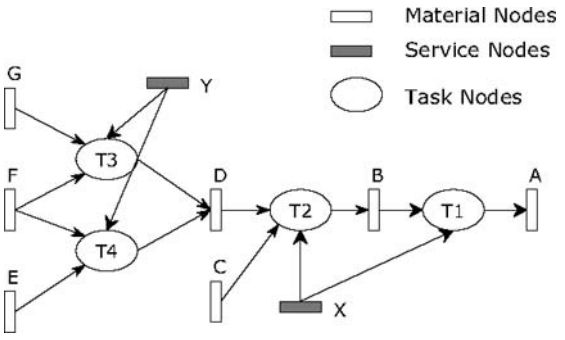

Fig. 5.8. Product recipe for illustrative example

5.3.2 Application of the DRPC Interaction Model

As per DRPC architecture, a product element PR is associated with product A. Each unit, header and service element is represented via a unit, header and service elements with symbols respectively as U, H and S. The (external) supplier elements for raw-materials are represented via prefix R. The process contains alternative process schemes through which materials A can be produced. These include U1, U2 in combination with any of the suppliers

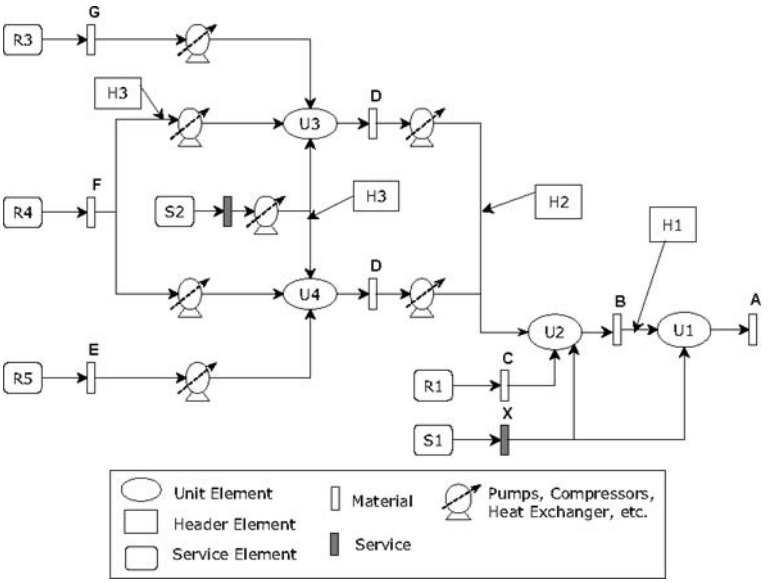

Fig. 5.9. Process layout for illustrative example

of D, *e.g.*, {U3}, {U4}, or {U3,U4}. The following sub-sections illustrate the sequence of interactions between product and unit elements using product-centric and unit-centric approach for recipe mapping (the header and service elements are omitted for simplifying the illustration).

Product-Centric Approach

Fig. 5.10 illustrates an animated sequence of interactions in the proposed model when a product-centric approach is used for recipe mapping. The individual steps therein can be described briefly as follows (the terms in brackets show the phase in interaction model to which the step corresponds to).

Step 0 : (*identify*) A new product element PR is created;
Step 1 : (*recipe mapping*) PR starts with announcing task T1 in recipe;
Step 2 : (*recipe mapping*) Since U1 can only execute T1, it replies back;
Step 3 : (*recipe mapping*) PR assigns T1 to U1;
Step 4 : (*recipe mapping*) Interactions repeat until all tasks in recipe are assigned;
Step 5 : (synthesis) U1 and then U2 send material requests to their supplier unit elements as identified in Step 4. The supplier elements return their proposals;
Step 6 : (*synthesis*) U1 and then U2 allocate their material demands to supplier elements. The configuration of process scheme is thus fixed;
Step 7 : (*reconfigure & operate*) The process scheme is reconfigured (as appropriate). Material and service flows are established;
Step 8 : (*terminate*) Process scheme is terminated when order requirements are met. PR is removed.

Unit-Centric Approach

Fig. 5.11 illustrates the sequence of interactions when a unit-centric approach is used for recipe mapping. Again the individual steps therein can be described briefly as follows.

Step 0 : (*identify*) A new product element PR is created;
Step 1 : (*recipe mapping*) PR announces its order requirement for producing product A;
Step 2 : (*recipe mapping*) U1 can produce A through task T1. It confirms with PR that T1 is allowed in product recipe;
Step 3 : (*recipe mapping*) U1 announces material request for B; U2 can supply B. It replies its interest;
Step 4 : (*recipe mapping*) U2 confirms its task T2 with PR and also extends the scheme to U3-U4;
Step 5 : (*recipe mapping*) U3 and U4 similarly confirm their tasks and extend the scheme;
Step 6 : (*synthesis*) U3-U4 and then U2 return supply proposals to their customer elements. U1 and then U2 then allocate their demands;
Step 7 : (*reconfigure & operate*) The process scheme is reconfigured (as appropriate). Material and service flows are established;
Step 8 : (*terminate*) Process scheme is terminated when order requirements are met. PR is removed.

Discussion

As can be seen from Figs. 5.10 and 5.11, the role of product element PR in the interactions is limited to that of assigning tasks (product-centric approach)

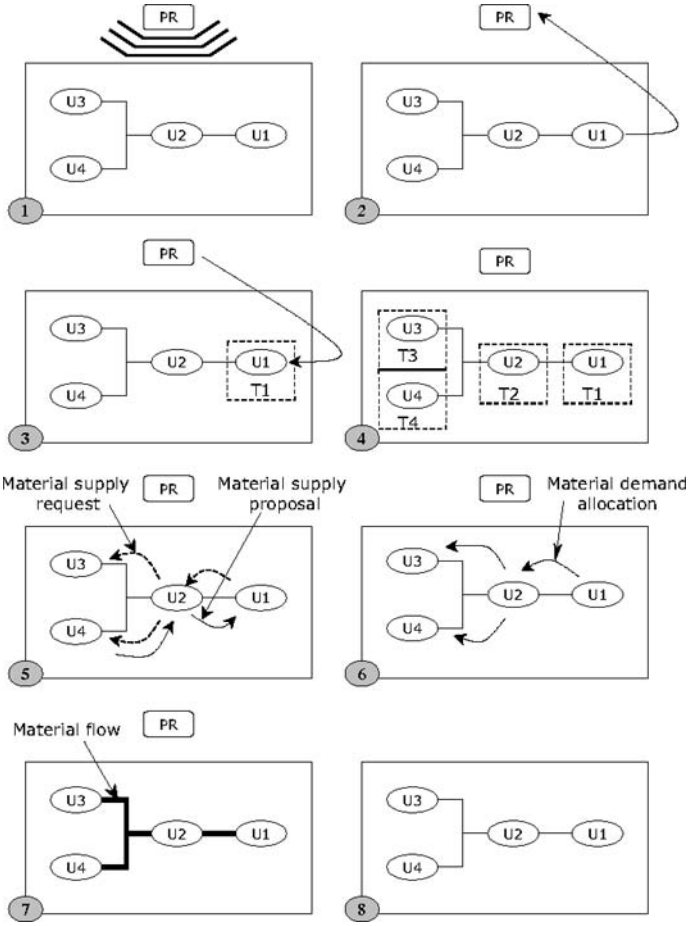

Fig. 5.10. Illustration of interaction model using product-centric approach

or confirming that the tasks selected by unit elements are allowed in the recipe (unit-centric approach). The actual synthesis of process scheme in terms of deciding the process parameters and local settings is carried out by unit elements themselves (together with header and service elements). As discussed in detail in the next section, this distinction forms a key difference in the proposed model compared to earlier interaction models in holonic or agent research.

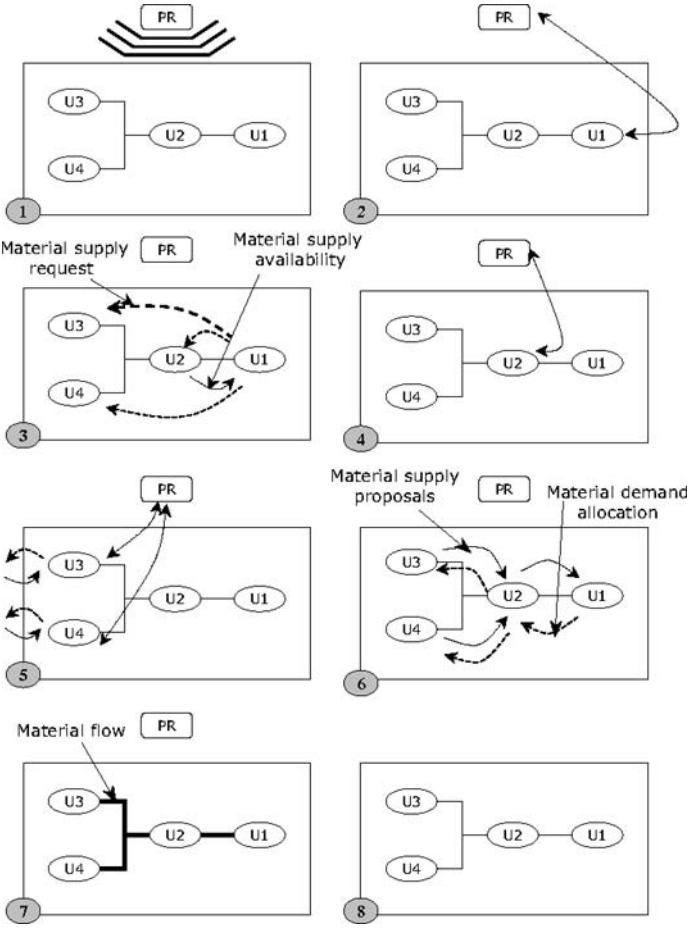

Fig. 5.11. Illustration of interaction model using unit-centric approach

5.4 Comments on the DRPC Interaction Model

To conclude this chapter, we comment on potential differences between the proposed interaction model and its analogs within existing conventional and alternative distributed approaches.

5.4.1 Comparison with Conventional Process Control

Conventional control is based on hierarchical information and control flow. As discussed in Section 2.3, the production in a hierarchical system is driven by a higher-level, long-term plan derived based on customer order forecasts (Williams 1989). Below, we first show that the interaction model described in this chapter is compatible to this conventional information flow in that all

control functions and interfaces that are implemented in a conventional system can also be implemented using the DRPC interaction model if need be. In addition, the bottom-up nature of interactions between elements offers several new benefits in the areas where hierarchical control is restricted. These are: (i) bottom-up response to change and disturbance, and (ii) graceful degradation of performance when failures occur.

Compatibility with Conventional Control

The DRPC approach decomposes the conventional hierarchy into localised control modules of process elements. Each element thus possesses a capability to plan, optimise and control its operations and also coordinate them with other elements. This suggests that by restricting the interactions of process elements to a limited set of process schemes and control configurations, one can implement the same control functionality of a conventional system using the framework of proposed model.

To illustrate this, Fig. 5.12 uses a simple example of level control in two series-connected tanks. The figure also includes two different control structures used frequently in conventional systems: (a) *control in the direction of flow*, where the product demand directly controls the flow of incoming raw-material, and (b) *control in the direction opposite of flow*, where the demand is propagated via level control in both tanks. The third scheme in the figure shows a DRPC approach operating in a demand-pull mode. In this case the variable pairings for each tank are combined and encapsulated into a general-purpose control module. By configuring this module as appropriate, the DRPC model can be made to behave as either of the two conventional schemes, because in either case the nature of interactions between elements in DRPC model remains the same *i.e.,* the demand information flows backwards and the variations in material flows forwards. A similar argument can be extended to other control levels to interpret the information flow at those levels in a demand-pull form. This indicates the compatibility of DRPC model to conventional control.

Bottom-up Response to Change and Disturbances

The example in the previous section briefly demonstrated the manner in which the process elements interact to provide a response to arrival of a new product order. The response emerges via bottom-up (*i.e.,* element-to-element) interactions. This behaviour is not predefined in the interaction model. A more detailed example illustrating these issues is given in Chapter 7. The model is thus able to deal with different scenarios or a combination thereof without an explicit definition of global response for each case.

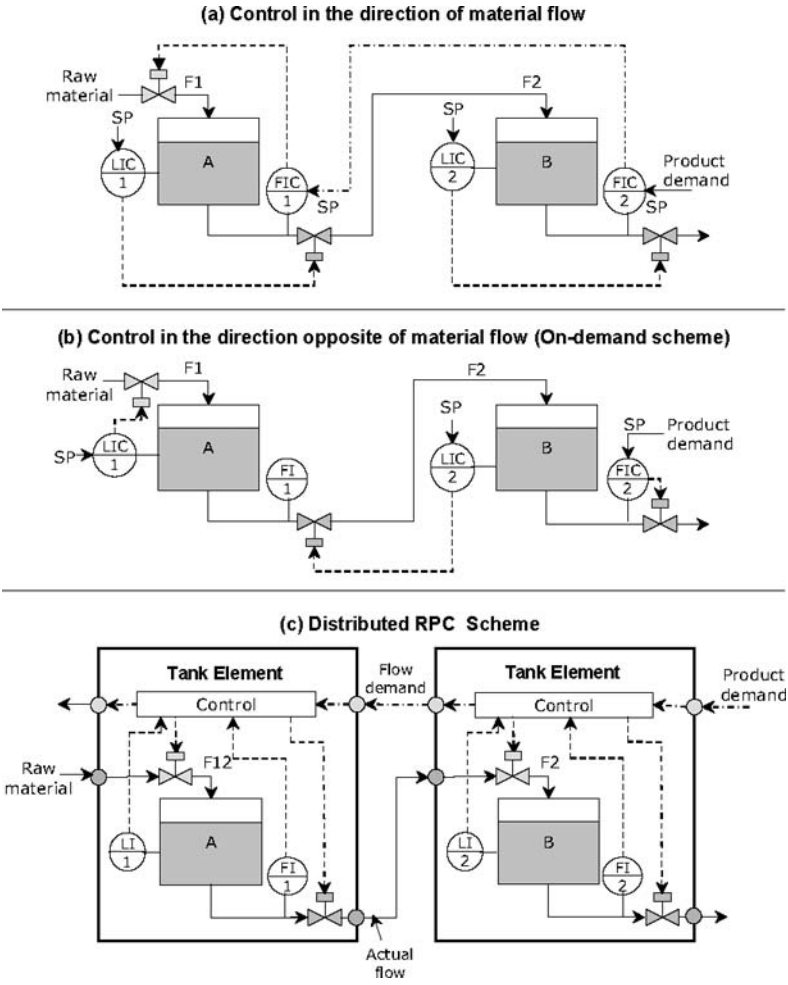

Fig. 5.12. Comparison of conventional and DRPC control structures

Graceful Degradation of Performance in Case of Failures

The interaction model provides a definitive structure and guideline on how should elements exchange information. To enhance the predictability of operations the model guarantees that the interactions are flexible but also binding. A unit element, for example, should not simply de-commit from supplying its products to downstream unit elements in case if its local process becomes bottleneck or one of its supplier unit elements fails. Rather, it seeks an alternative supplier element for the same feedstock. Only under the circumstance where the disturbance is sufficiently large (in magnitude or of long-term nature), it opts to propagate the disturbance further or terminate its processing tasks.

The interaction model thus ensures that performance degrades gracefully until a point is reached where the disturbance can be fully absorbed.

5.4.2 Comparison with Other Distributed Interaction Models

As noted in the introduction, the principle of contracting has been the basis of research in most previous holonic or agent research (Gou *et al.* 1998, van Brussel *et al.* 1998, Chirn & McFarlane 2001). The proposed model extends contracting or market programming approaches by using a virtual enterprise based model so as to address the constraint of physical connections between process units. The model thus differs from the previous research in various ways.

The Role of Product Elements

In the proposed model, the product elements interact with unit and header elements to map the processing tasks onto production capabilities. Unlike other architectures though (*e.g.*, PROSA (van Brussel *et al.* 1998) or its related architectures), the product elements do not manage the logistics of materials or services in the network nor do they define the operating conditions of unit, header or service elements. Such decisions are made by these latter elements themselves once the processing tasks are assigned. This modification hence avoids the complexity of coordination if the product elements are allocated with this responsibility. Additionally, as it was shown with unit-centric approach, the distribution of recipe mapping provides unit elements with an increased freedom to select local tasks that best match with the current plant status.

Network Behaviour of Process Elements

As can be seen, the virtual enterprise paradigm provides an effective approach to contracting or market approaches in dealing with the physical constraints. The unit, header and service elements, for instance, are now made able to interact with: (a) other such elements on process connections, and (b) product elements on product-recipe related issues. Product elements instead behave purely as information servers monitoring the adherence to product recipes. Moreover, as mentioned in Section 4.5, the whole network operation is decoupled by the use of header elements, that similar to transporters in supply chains, can be made flexible as necessary (by adding extra piping streams or transfer equipment equipment) irrespective of other elements requiring their use.

The proposed model also operates on a demand-driven basis, *i.e.,* the unit elements build processing schemes (in the form of material and service exchange protocols) in backward direction starting from the end-products to

raw-materials. Subsequently, any new demand changes imposed on the process are also propagated in the process in an incremental manner. This behaviour ensures that the production remains fitting to changing demands.

The use of backward search in building or extending process schemes (see Fig. 5.10 and Fig. 5.11) also guarantees that the resulting partial schemes are feasible, *i.e.,* physically implementable. This may not so with other architectures mentioned above where the interactions between product and resource elements generally follow a dispatching mode of task allocation, *i.e.,* the next task in sequence is only announced when the previous task is finished. This could possibly result in dead-locks and dead-ends where the product elements may find no further machine available to progress the partly finished parts. The proposed model avoids this scenario by ensuring that the unit elements (together with header and service elements) build a complete process scheme from raw-materials to end-products before the actual production commences.

Distribution of Information and Control Functionality

Unlike contracting, the proposed model also improves the distribution of information and control among elements as the product elements are no longer responsible for coordinating local operations. Hence, little or no production information (depending on the product or unit-centric approach used for recipe mapping) needs to be transferred to product elements. This feature can be of significant use when: (a) the number of product elements that can coexist in the process is large; (b) the product elements are designed and developed by teams situated remotely, or (c) multiple product elements share some of the materials or unit elements which can lead to deadlocks because the supplies of materials or these shared unit elements are likely to fail.

5.5 Summary

In this chapter we have proposed a distributed interaction model to support the run-time interactions of process elements in the control architecture. We next go onto examining the quantitative aspect of these interactions, *i.e.,* to define a distributed solution strategy for use of the process elements, in particular the unit elements, to identify their local operating settings during the define, reconfigure or operate phases.