# 3

# DRPC: Distributed Reconfigurable Process Control

## 3.1 Introduction

In the previous chapter we noted that a distributed approach to the design of a process control environment may provide a route to increased reconfigurability and with that the associated business benefits. In this overview chapter and the following three chapters we begin to build up a blueprint for how such a distributed control system might be constructed. We have already noted that such an approach is – conceptually – fundamentally different to conventional, hierarchically-based control systems and because of this we begin with a recasting of the process control system structure before moving on to examine the way such a system might operate.

In this chapter we first gather together the needs for a reconfigurable process control system – as discussed in Chapter 2 – and then marry these with the notions of distributed coordination as defined from the related but different developments described in Section 2.5. This allows us to produce a conceptual overview of how a distributed and reconfigurable process control system might operate. From here, we identify the key developments needed in order to construct such a system, namely, the architecture and element designs, the interaction between distributed elements and the governing optimisation strategy for achieving globally well behaved control.

## 3.2 Addressing the Needs for Reconfigurable Process Control

In the solution we are going to develop we seek to address the needs for reconfigurable process control as identified in Section 2.3.3. These were summarised into four system requirements: (i) product and process diversity, (ii) easy modifiability, (iii) responsiveness to change and disturbances, and (iv)

fault-tolerance with graceful degradation of performance (Fig. 2.7). The state-of-the-art review of research in process control showed that each of these requirements have been addressed – but only individually and only for rather limited class of applications. A holistic approach addressing them within a single framework is yet to appear. We propose here that a distributed coordination approach based on holonic principles can provide one such approach. The previous results in holonic research, however, apply to discrete manufacturing and therefore do not translate straight to process control. So, to develop a distributed RPC approach, we need to understand what opportunities and constraints exist in process control that are different to discrete control, and from that, decide how to adapt the existing holonic research.

### 3.2.1 The Reconfiguration Process

Before assessing how to use the existing research to address the needs of reconfigurability, we firstly examine the process of reconfiguration itself. Fig. 3.1 outlines all the key steps in the reconfiguration of a complex system which range from the identification of an opportunity to the coupling/recoupling of elements, the reorganisation of those elements and the monitoring of the outcome. First we concentrate on the central section: that of effecting the reconfiguration process. Intuitively, this can be split into three actions: (i) *decouple* – pull apart system elements from existing configuration, (ii) *reorganise* – reorganise them into new configuration, and (iii) *recouple* – put them together to operate. All three actions may involve a physical change (physical set-up of the process units or their interconnections) and/or a control change (operating settings, recipe parameters *etc.*). The reconfiguration may be requested as planned (*e.g.*, introduction of a new product order) or unplanned (*e.g.*, failure of a process unit). An RPC system capable of tackling these conditions shall provide the support necessary to carry out the above actions smoothly and efficiently, and preferably in an *automated* fashion. In addition, the RPC system must also *identify* that a reconfiguration is necessary from *monitoring* of the plant conditions or the arrival of new production opportunities (new orders *etc.*), and *define* the structure of the new configuration. In conventional practices, these actions would be performed offline (when planned) or by human intervention (when unplanned). However, in more dynamic scenarios that can arise in future, such methods may fail to cope with the complexity and fast timescales that may be demanded. Instead, we believe these actions shall as well be included as part of the scope of the RPC system with, if necessary, an extra level of automation and intelligence provided that induces self-reconfiguration.

### 3.2.2 Adapting Existing Holonic Systems Research

The previous holonic research has used distributed interactions between holons to manage the reconfiguration process in a manner described above.

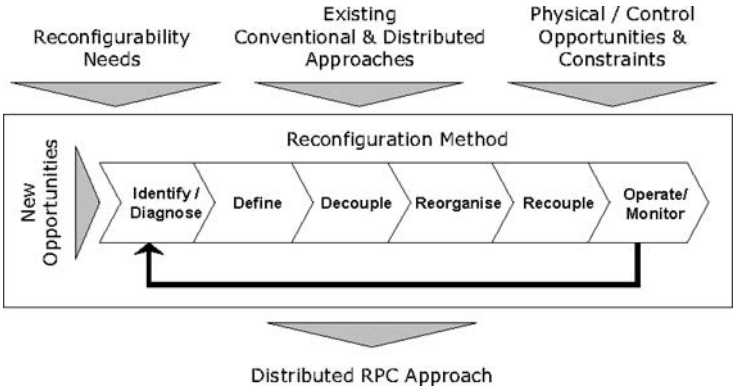


**Fig. 3.1.** Research approach

Generally, a contracting protocol (Smith 1980) or its extended variants such as based on lagrangian decomposition (Gou *et al.* 1998) or market programming (Váncza & Márkus 2000) are used to define the information structure for interactions between holons (Fig. 3.2(a)). In contracting the interactions occur between product and resource holons. In a so-called *task-based view* of contracting, the product holons act as the *managers* and distribute tasks to appropriate resource holons. In turn, they also coordinate the flow of parts across shopfloor. In a dispatching mode of operation, the interactions build progressively with new tasks only assigned when the previous tasks have finished. The resource holons themselves are assumed physically decoupled (in jobshops) or connected via buffers such as storage units and conveyor queues that are assumed to decouple their operations (in flowshops). The resource holons therefore do not interact directly or coordinate their operations. In an alternative *resource-based* view, the resource holons instead act as the managers and announce their availability and accept tasks that best suite the local conditions. Again the interactions occur between product and resource holons with the former acting as the coordinators of the flow of parts.

In a continuous process the situation remains different though because the process units remain tightly connected via piping streams and their local dynamics interact in most cases due to lack of interim storage or buffer tanks. That means, the processing tasks assigned to any process unit must match – in a physicochemical term – to that of its neighboring units. A tight coordination of unit operations is thus essential, both at the time of allocating tasks to process units and also when these tasks are being executed. In most cases it may also be necessary that the entire task sequence in the product recipe, *i.e.,* from raw-materials to end-products, is developed first before the execution of the first task can start; a dispatching mode of task release and assignment cannot simply work. In these conditions if the above task-based view is used for managing interactions, then it is likely that the coordination

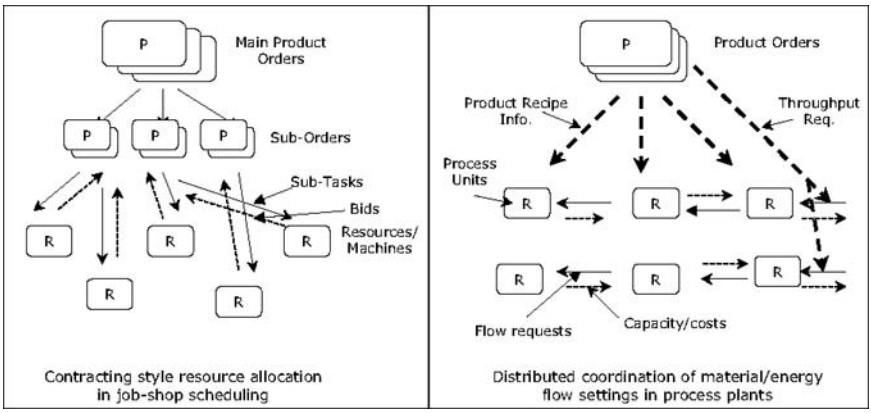


**Fig. 3.2.** Distinction between: (a) contracting-style resource allocation in discrete manufacturing, (b) proposed form of distributed coordination in process plants

effort required to satisfy these additional constraints can become excessive due to centralised role of product holons and the chances of potential conflicts between their task assignments. Instead, a distributed method for RPC must be better off by achieving coordination through direct interactions between production functions themselves (*i.e.,* resource holons) whilst also interacting with product functions (*i.e.,* product holons) where necessary. This distinction is clarified in Fig. 3.2(b).

### 3.2.3 Analogy from Virtual Enterprise Management

A useful coordination method for a DRPC approach can be developed by extending the contracting principle in previous research with an analogy from supply chain management. In particular, we borrow an analogy from so-called *virtual enterprises* or *dynamic* supply chains as discussed earlier in Section 2.5. In a virtual enterprise multiple equal-interest companies come together to form a temporary alliance that delivers the fast-changing customer demands. The alliance evolves in time and adapts to changing marketplace when necessary. The reconfiguration of the whole chain, including that of coordinating the material flows, occurs via direct, distributed interactions between companies themselves. A framework analogous to these interactions of companies can be considered to define the interactions of product and production functions within an RPC system.

In particular, we consider an analogy from the 'life cycle model' of a virtual enterprise. A virtual enterprise normally goes through four main phases during its life cycle as shown in Fig. 3.3: (i) identification, (ii) formation, (iii) operation, and (iv) termination (Strader *et al.* 1998). The identification phase starts with a research of available market opportunities and identifying an opportunity that can be pursued further. This is then input to the formation

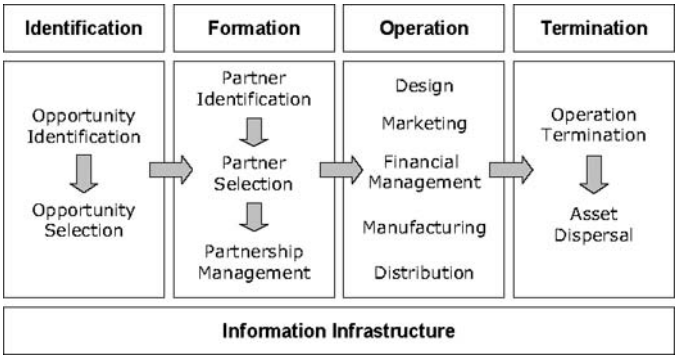


**Fig. 3.3.** Life-cycle model of a virtual enterprise (*Source:* Strader *et al.* 1998)

phase. The formation involves identification and selection of partnering companies and building from that a chain that can deliver the order requirements. Different combinations of partners and business processes may be evaluated before the companies arrive at a choice of the configuration. The selected partners then integrate their business processes during the operation phase in order to deliver the order. During this they may exchange local information such as stock levels, demand forecasts *etc.* to improve their visibility across the chain. The network is finally dissolved and the shared assets, if any, are dispersed or re-used to initiate a new opportunity.

In an analogous manner, the production functions in the proposed DRPC approach use distributed interactions to manage the reconfiguration process in Fig. 3.1. An outline of how this analogy would operate is illustrated in Fig. 3.4, which compares the order fulfillment process in a virtual enterprise and in a DRPC system. In a virtual enterprise, the evolving market opportunities drive the partnering companies to come together and form alliances while in a DRPC system, the customer orders (or so-called *product elements*) drive the different production functions to configure appropriate process schemes and deliver the order requirements.

The analogy, while conceptually sound, is faced with certain challenges. Process plants, unlike supply chains, are characterised by shorter time-scales, the non-linear and dynamic behaviour of process units, and the material and energy recycles – the features which are not normally critical in supply chains. However, if these complexities are taken aside and if the processes are seen purely as chains of material and energy flows, then it is possible that the network behaviour of process units, *e.g.*, in a multiproduct or multipurpose plant, can still be examined in a manner similar to that of companies in a virtual enterprises. The analogy in this sense can provide an useful aid to visualise the operations of distributed production functions in a process plant in a manner much similar to the use of contracting in previous holonic research.

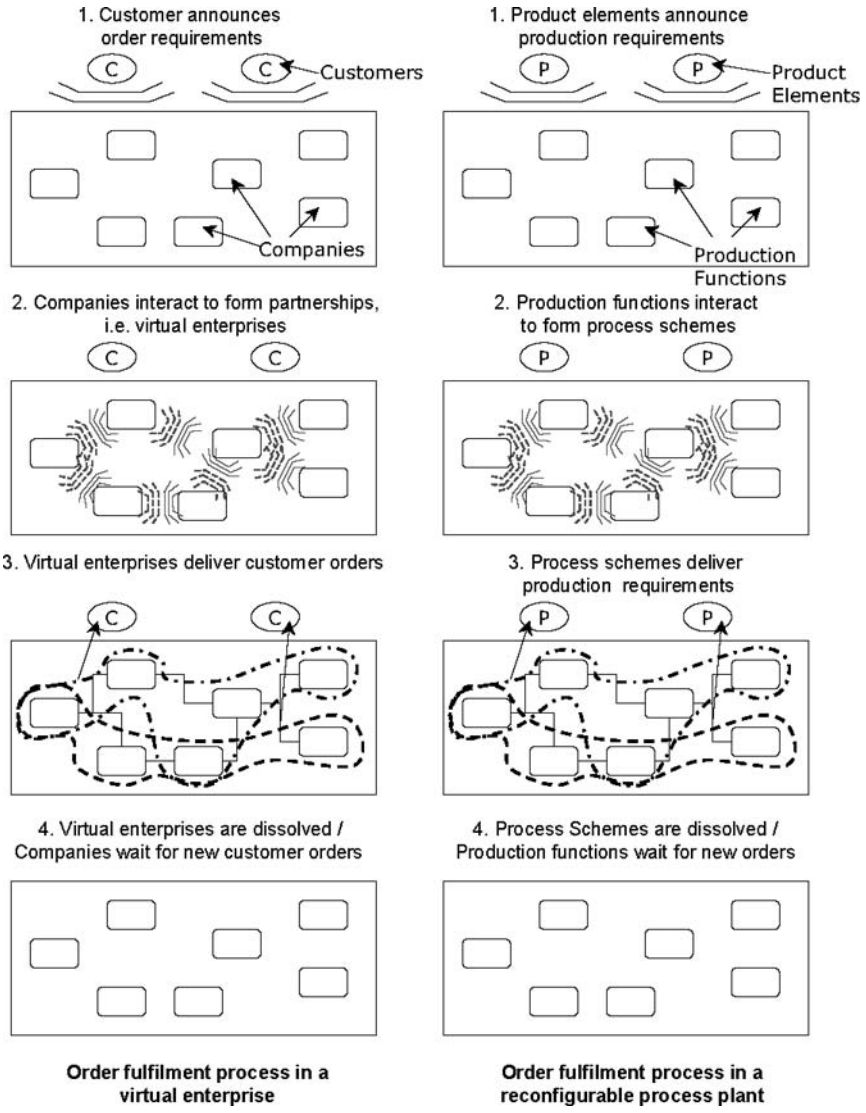


**Fig. 3.4.** Order fulfillment process in a virtual enterprise and a DRPC system

## 3.3 Introducing the DRPC Approach

In line with the previous research in holonic and agent-based industrial control, we now begin to develop the DRPC approach by describing the tools necessary for constructing an RPC system. The method of *top-down decomposition* and *bottom-up integration* is considered the key to these developments. In particular in the following chapters we focus on the following three areas.

i. *Distributed Control Architecture:* A control architecture is developed in order to characterise the different production functions (so-called *process elements*) and their primary control responsibility.

ii. *Distributed Interaction Model:* Next the structure of information exchange between process elements is defined so as to cover the interactions necessary for implementing the reconfiguration process in Fig. 3.1.

iii. *Distributed Control Strategy:* To support their reconfiguration decisions, the process elements are finally supplied with control strategies in the form of a distributed algorithm. Only a generic problem is investigated at this stage so as to complete the architectural description.

The following chapters in this part of the book address each of these developments in turn.