4

Reconfigurable Process Control Architecture

4.1 Introduction

We now begin to construct the DRPC system from its basic elements and specify how these elements are defined for typical process control functions. The connection between these elements – the so-called control architecture – defines the structure for the process control system resulting from their combination.

4.1.1 Overview

We start with understanding the terms *architecture* and *control architecture*, *i.e.*, what properties should a control architecture have and what are its key elements.

The term *architecture* can be defined broadly as the attributes of a system as seen by its designer, or formally as, a conceptual structure of the system that also defines its functional behaviour while being distinct from the detailed design and physical implementation (Amdahl, Blaauw & Brooks 1964). An architecture in this sense forms a critical input to the design process to lay down the specifications of end-user requirements based on which the actual system can be built.

Over the years, two different meanings of 'architecture' have evolved in systems engineering: (i) the architecture as a generic 'style' or a 'method' for building one or more systems, called the *reference architecture* and (ii) the architecture as a 'product' or a 'template' for a specific system, called the *system architecture* (Williams 1989, Zwegers 1998). The reference architecture sets out the generic behaviour and attributes and possibly the rules of design for a number of similar systems. A system architecture instantiates the behaviour and attributes of the reference architecture by applying these rules to a specific application. Fig. 4.1 outlines the use of reference and system architectures within overall systems engineering process.

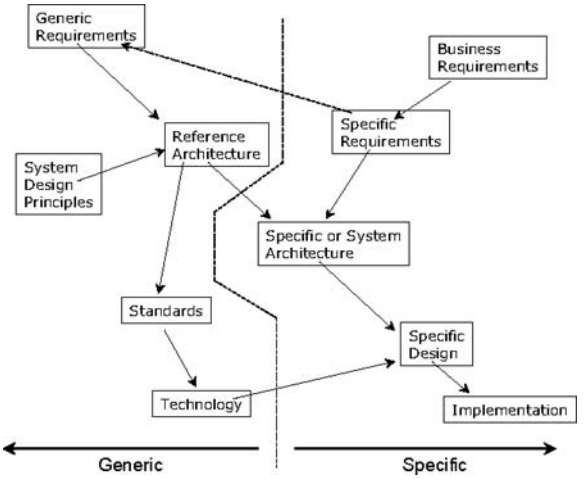

Fig. 4.1. Role of architectures in systems engineering (*Source:* Williams T.J. 1989)

The term *control architecture* refers to an architecture of a manufacturing control system (or in the context of this research, for a process control system). We limit it to be a reference architecture in this text. The role of a control architecture in this regard is to allocate the various decision-making responsibilities for production control to the specific control components or controllers. Further, it should determine the relationships between these controllers so as to establish a mechanism for coordinating the execution of their decisions (Dilts, Boyd & Whorms 1991, Senehi, Wallace & Luce 1992).

The research on manufacturing control architectures has evolved over the years. Historically, the early architectures defined as part of Computer Integrated Manufacturing (CIM) were hierarchical – so-called 'proper' hierarchical as Dilts *et al.* (1991) call them. Some key examples include AMRF (Jones & McLean 1986) and MSI (Senehi *et al.* 1992) in discrete manufacturing and Purdue Reference Model (Williams 1989) in process industry. Hierarchy helped manage the size and complexity of control functions that the earlier centralised structures failed to handle. But as the time progressed, it was recognised that hierarchy can have its own shortfalls. The inflexible structure of hierarchy due to multiple levels of control and the delays in passing information between these levels could result in poor response to unforeseen change and disturbances. To overcome these shortfalls so-called *heterarchical* or flat architectures were proposed as comprising distributed, locally autonomous controllers without any master/slave relationships (Duffie & Piper 1987, Duffie & Prabhu 1994). The benefits of hierarchy and heterarchy have been long debated as heterarchy can result in chaotic performance due to lack of coordination where hierarchy was shown to perform better (Bongaerts, Monostori, McFarlane & Kádár 2000). *Holarchy*, a term coined as part of holonic research

(Koestler 1967, Christensen 1994), is considered to deliver the benefits of both hierarchy and heterarchy whilst also avoiding their shortcomings. Unlike hierarchy, the system elements in a holarchy remain distributed, loosely-coupled, but unlike heterarchy they also coordinate their operations across the plant. They can behave both as pro-actively and reactively under different conditions that in a way enhances their reconfigurability. We exploit this dual property of holarchy in defining the behaviour of process elements in the control architecture to be developed in this and later chapters.

4.1.2 Requirements for the RPC Architecture

In this chapter, we aim to develop a control architecture for RPC systems with a focus on so-called semicontinuous class of process systems. The architecture is expected to help at least meet the requirements from Fig. 2.7 of product/process diversity and easy modifiability as they both heavily depend on the architectural properties of a process control system. In addition, it should help improve responsiveness and fault-tolerance of the system by ensuring that constituent control elements of the architecture are sufficiently decoupled and that the propagation of disturbance or failures across the system remains limited or occurs gracefully.

This chapter is structured as follows. Section 4.2 next describes the structure, data models and basic control functions of distributed process elements in the architecture. An incremental approach to migrating to this fully distributed form of control is suggested in Section 4.3 so as to allow industrial practitioners to experiment with these new concepts using existing off-the-shelf control tools. Section 4.4 applies the architecture to an example polymer process plant. Some comments on the architecture in terms of the above mentioned requirements and the other conventional and distributed architectures are presented finally in Section 4.5.

4.2 Specification of Process Elements in a RPC System

We now describe the specification of distributed process elements in the proposed RPC architecture. Following the approach described in Section 3.2, we consider a supply chain (in particular virtual enterprise) based analogy to visualise the structure and behaviour of elements in the architecture.

4.2.1 Basic Types of Process Elements

The proposed architecture divides the functionality of a process control system into four primary types of process elements, called: (i) unit element, (ii) piping header element (in short, header element), (iii) service supplier element (in

short, service element),[1] (iv) product element. The functionality is divided based on physical structure of the process instead of temporal or multi-level decomposition as in a hierarchical system.

The functionality of these individual types of process elements in the architecture can be defined as follows:

- *Unit Element:* A process unit element (in short unit element) represents a physicochemical processing task such as reaction, distillation *etc.* in the process. The task may have associated with at least one but possibly more control decisions that the unit element can regulate on its own.

- *Header Element:* A header element represents the logistics of materials or services within a specific segment of the overall process network. Physically, it may contain a number of piping streams, transfer equipment (pumps, compressors *etc.*), final control elements, energy transfer units (heat exchangers *etc.*) and storage units. These component sub-units should not incur any physicochemical operation, however they can be used to change the physical state of the material or service being transferred, *e.g.*, heat, cool, pressurise or depressurise them.

- *Service Element:* A service supplier element represents a custodian responsible for allocating a service to customer process elements that use this service in their local tasks. The customer elements can be either unit or header elements. Multiple service elements may exist in the process, each supplying one or more different services.

- *Product Element:* A product element represents the production requirements of a specific customer order in the form of a product recipe (specifying the sequence of processing tasks to be used or allowed) and other requirements such as quality, quantity and throughput of the product demand. Multiple product elements may co-exist in the process, each representing a specific customer order, but only a few may be produced at a time. Note that unlike the previous three elements, the product element does not have a physical presence in the process; it only acts as an information component supplying necessary product information to other process elements.

Fig. 4.2 depicts examples of various unit, header and service elements that can be found in process industries. Important to notice is that the header elements decouple the operations of unit and service elements in a sense that the physicochemical tasks of unit elements or the service supply tasks of service elements can be identified and defined more clearly and separately from the

[1] The term *service* refers to utilities such as steam, cooling water, electricity and other such enabling facilities (for example manpower) that are used in the execution of various processing tasks and the transfer of materials.

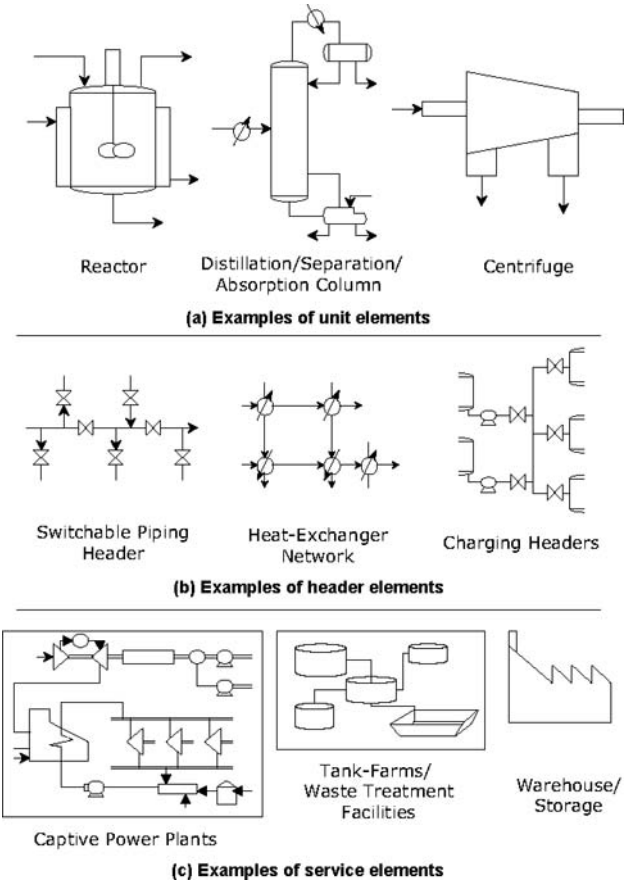

Fig. 4.2. Examples of unit, header and service elements in process industry

transfer/transform tasks of header elements; the header elements can thus be made flexible as and when necessary by adding extra transfer facilities without having to modify the interface of unit or service elements.

We note that the above identification of such element types is not strictly new to the distributed coordination field. Except for the service element, the notions of unit, header and product elements have previously appeared in vivid forms as so-called resource, transport and product holons in other distributed architectures in holonic and agent research (*e.g.*, PROSA (van Brussel *et al.* 1998), HCBA (Chirn & McFarlane 2001), HSCF (Cheung, Yeung, Ng & Fung 2000), ADACOR (Leitão & Restivo 2006)). However, as explained later in this chapter and the next chapter, the roles and interactions of process elements are different in the current architecture than these previous architectures. The differences primarily emerge due to the physically distinct nature of operations

Table 4.1. Analogy between supply chains and reconfigurable process plants

Process Plants	Supply Chains
Unit elements	Echelons (manufacturers, retailers *etc.*)
Header elements	Logistics providers (transporters, storage units *etc.*)
Service elements	Facilitators (investors, banks *etc.*)
Product elements	Customers

in a continuous process then in a discrete manufacturing process discussed later in this chapter.

The concept of service element is specifically new to this architecture. It relates to process enterprises where a number of plants or process units situated next to each other share common services (steam, cooling water, raw-materials, *etc.*) supplied by separate supplier facilities (captive power plant, cooling water plant, *etc.*). It is widely known that an effective distribution of common services can prove to be significant at times when the supplier plants fail or the supply-demand balance of services is disturbed for some reasons. At other times when conditions are planned, an optimal distribution can increase company profits substantially. The role of a service element, being a custodian of one or more services, is to interact with the respective customer elements so as to coordinate the distribution of its services in a manner that is effective and responsive at times.

We note in passing that the above identification of four process elements is also related to their analogous components in supply chains and in particular virtual enterprises. Table 4.1 shows this link, which suggests that if a process plant is considered a form of (mini-)supply chain, then the unit elements are the echelons in the supply chains, the header elements are the logistics providers, the service elements are the facilitators or service providers, and the product elements are the final customers. The analogy thus provides a systematic, ontological concept (as an extension to the contracting principle in previous holonic or agent research) to define the interactions of process elements. This is discussed in more detail in the next chapter.

4.2.2 Data Model and Control Functions of Process Elements

All four process element types possess associated roles, data models and control functions in the architecture. These can be described as below. This information then forms a part of the interactions of elements in the next chapter:

- *Unit Element:* The role of a unit element is to perform one or more processing tasks. To satisfy this role, it executes the following functions: (i) identify the processing tasks it should perform by interacting with respective product elements and other unit elements; (ii) acquire necessary feedstocks and services for these tasks from respective supplier elements;

and, (iii) perform the processing tasks to convert incoming feedstocks to outgoing products. Depending on the properties of incoming feedstocks and the specification of outgoing products, the exact tasks that a unit element performs and the type of services it requires can vary time-to-time.

- *Header Element:* The role of a header element is to transfer one or more materials or services within a segment of the process network. To satisfy this role, it executes the following functions: (i) identify the configuration of the process routes through which the materials or services are to be transferred; (ii) identify the requirements for property change for the materials or services being transferred (*e.g.*, heat, cool, pressurise, depressurise them); (iii) develop and implement a procedure(s) to switch the process routes from their current configuration to required target configuration; and, (iv) transfer materials or services in a controlled manner by interacting with respective unit or header elements.

- *Service Element:* The role of a service element is to distribute one or more services to its customer elements. To satisfy this role, it executes the following functions: (i) identify the nature of service demands from customer elements and decide the service supplies available for those demands; (ii) determine an optimal, or when necessary an emergent but sub-optimal, distribution of services while taking into account the priorities of service demands; and, (iii) distribute the services in a controlled manner via interacting with respective customer elements.[2]

- *Product Element:* The role of a product element is to represent the requirements of a production order and to ensure that these are met. To satisfy this role, it executes the following functions: (i) identify the processing tasks to be executed; (ii) map these tasks onto production capabilities of unit and header elements available in the plant; and, (iii) engage with unit and header elements to allocate these tasks (see next chapter for more details on how this mapping and allocation of tasks is carried out). Unlike previous distributed architectures and as discussed in Section 3.2, the product elements do not directly coordinate the operations of unit or header elements or the flow of materials or services in the network; this is done by unit, header and service elements themselves via direct interactions.

Fig. 4.3 depicts an UML diagram (Unified Modelling Language) of the data model and control functions of all four element types. The *association* relations, shown by solid lines, denote the presence of interactions between

[2] A service element may comprise its own internal production system to produce services. This process can be similarly represented via appropriate unit, header and service elements. When referred to the main system, the service element then also acts as a type of product element representing the composite demands of customer elements requesting its services.

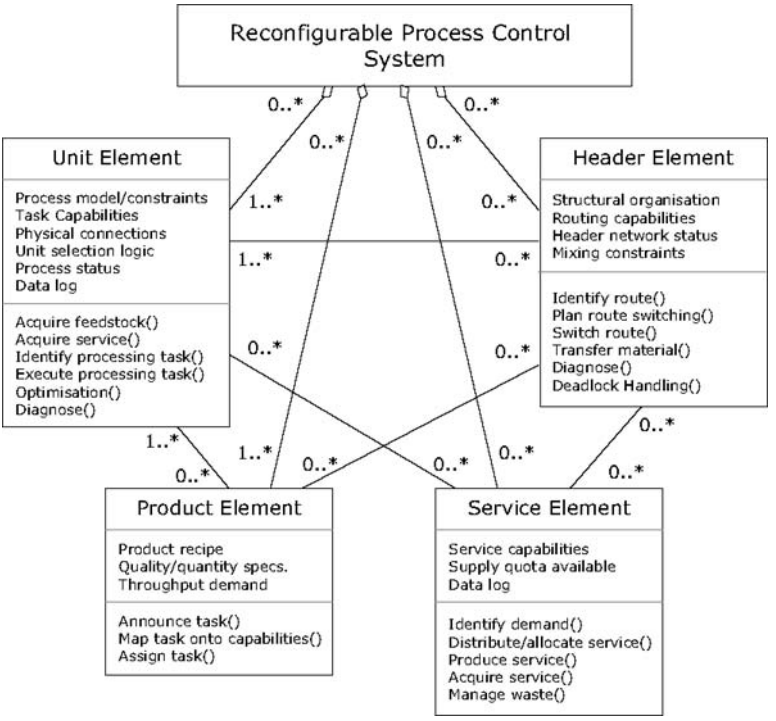

Fig. 4.3. Data model and control functions of process elements

process elements while the *aggregation* relations, shown by solid lines with diamond heads, denote the aggregation of process elements into an RPC system (Booch & Rumbaugh 2005). As the multiplicities in the figure suggest, a continuous process based on an RPC system must have at least one unit element and one product element in order to be able to produce a product. As the complexity of the process grows with more unit elements included and these elements sharing various services, the architecture requires adding further header and service elements.

4.2.3 Internal Structure of Process Elements

Internally, each process element is considered a self-contained control function comprising its local control module, a co-ordination module and the associated (optional) physical process part as shown in Fig. 4.4.

The internal design is derived initially from a decomposition of the multi-level control hierarchy as illustrated in Fig. 4.5. Each layer in the hierarchy is split along the physical dimension, followed by integrating vertically the localised blocks into control and coordination modules of process elements. The control module in Fig. 4.4 covers the execution functions (*i.e.,* basic

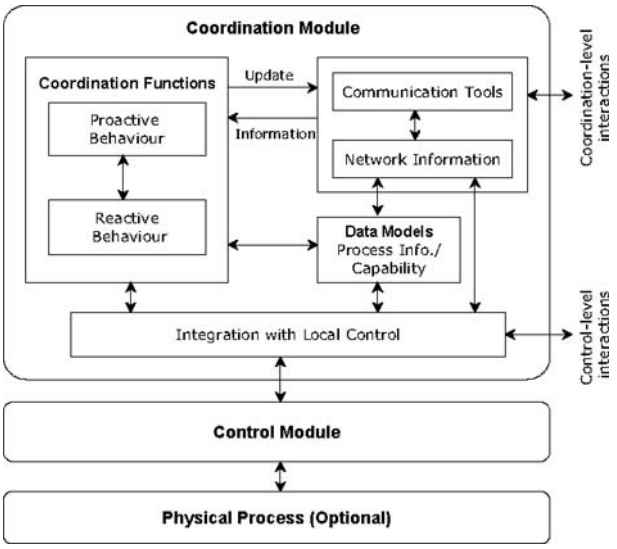

Fig. 4.4. Internal structure of process elements

control and some advanced control functions) and the coordination module the decision functions (*i.e.,* advanced control and levels above in Fig. 4.5). In addition to this, new components are included within coordination module to define the data models (process structure, capability *etc.*), the coordination functions (proactive and reactive behaviour) and the communication means to interact with other elements. Each process element thus receives the ability to plan, optimise and control its operations plus that of the relevant global system by coordinating with other elements.

4.2.4 Physical Connections Between Process Elements

Because of the manner in which basic element types are identified (*i.e.,* based on physical structure of the process instead of functional hierarchy), the process elements remain connected via material and service streams at process level. Fig. 4.6 depicts the five categories of such connections.

- *Material flow between unit elements:* This flow leads to production of the end-products. The flow may occur on a forward path (from raw-materials to end-products) or on a recycle path (from recovery units back to upstream units or intermediate storage).
- *Service flow to unit elements:* This flow may be required for the execution of processing tasks of unit elements (*e.g.*, supply of steam or cooling water for a reaction task).
- *Service flow to header elements:* This flow supports the transfer of materials or allows changing their properties, *e.g.*, heat or cool them.

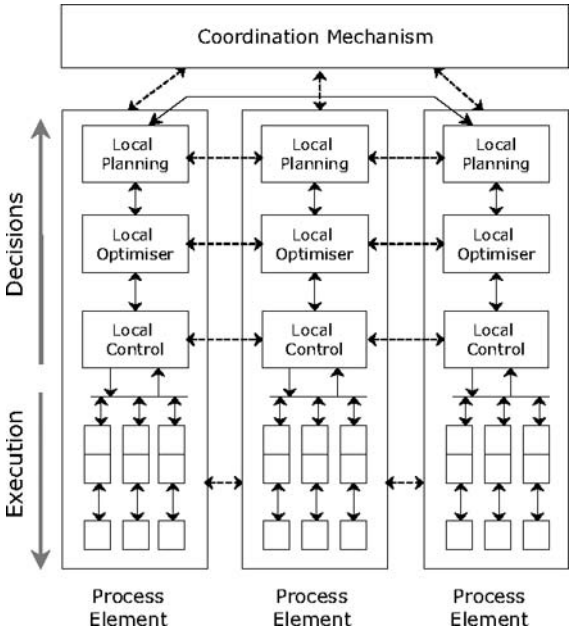

Fig. 4.5. Decomposition of multi-level control hierarchy

- *Exchange of services between unit and header elements:* This flow refers to recovered services from unit elements to be reused in the process (*e.g.*, heat released by exothermic reactor can be used to heat other materials)
- *Exchange of services between header elements:* This flow refers to recovered services from header elements to be reused in the process.

Note that the product elements do not have a physical presence and are not shown in Fig. 4.6 or described here. Their role is to provide unit and header elements with the product recipe information and they do so at the coordination level.

4.3 Migrating to Process Elements

The identification and design of process elements, being of fully distributed form, can be a radical change to the design practices currently in use in the industry. In order that these new concepts can be experimented – at least partially – using the commercial off-the-shelf tools available in DCS and PLC architectures, we can consider a migration approach based on an incremental decomposition of the control hierarchy. Previously, such an approach was also suggested by Chirn & McFarlane (2001) in the context of a discrete manufacturing architecture.

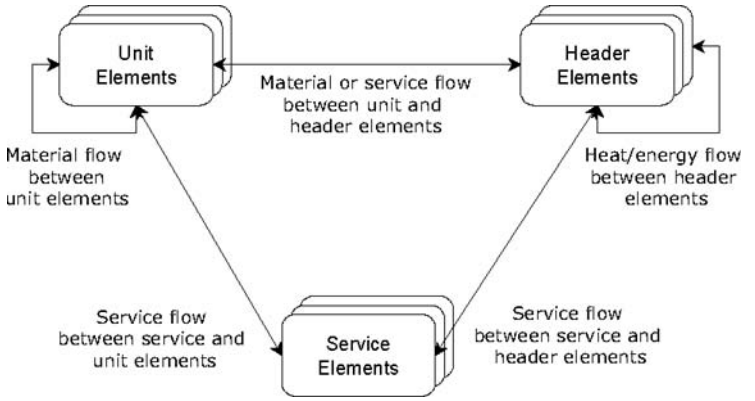

Fig. 4.6. Physical connections between unit, header and service elements

As a first step to migration, it is proposed that only the individual levels in hierarchy are decomposed as shown in Fig. 4.7(b); Fig. 4.7(a) shows the existing structure. These individual levels can still be implemented separately as in a conventional system using commercial tools of today. The individual levels may require developing a separate problem solving mechanism to enable them solve the control problems in a distributed form.[3] Next, one or more levels in the hierarchy in this distributed form should be integrated vertically (*e.g.*, optimisation and advanced control) so as to distribute more of higher level decisions down to lower levels (Fig. 4.7(c)). Finally, all levels in the hierarchy should be integrated vertically (Fig. 4.7(d)) such that the decisions requiring coordination are made by the coordination modules of elements and the execution of the outcomes of decisions is carried out by the control modules. It is envisaged that these vertically integrated design can be *packaged* together with respective physical process parts of process elements and supplied as stand-alone components to be plugged into an RPC system.

4.4 An Illustrative Example

We now apply the proposed architecture to a polymer process example shown in Fig. 4.8. The purpose of the example is to illustrate the selection and basic functions of process elements within an industrial process.

4.4.1 Description of the Process

The example process comprises two independent production lines, each comprising two polymerisation reactors. The process starts with reaction between

[3] In Chapter 6 a method to achieve this form of distribution is developed for a simplified control problem relating to the optimisation or advanced control levels.

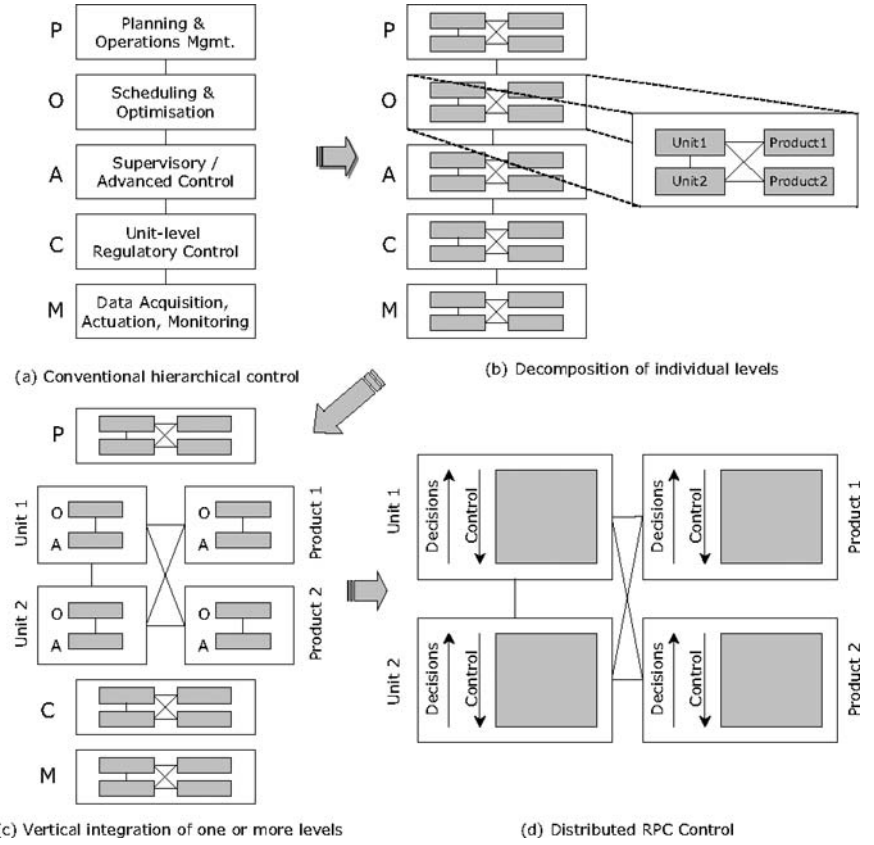

Fig. 4.7. Migration approach for developing internal designs of process elements

two main raw-materials (monomer and demineralized water) in the presence of other ingredients. This results in a slurry form of the end-product containing un-reacted raw-materials. The un-reacted raw-materials are separated in the flash vessels and stripper columns while the purified product is dehydrated and dried in the centrifuge and drier units before sent for storage. The recovered monomer is compressed and cooled for reuse again in the main reaction. The whole process operate in a batch-semicontinuous type, *i.e.,* the reactors operate in a batch mode while the other units in a semicontinuous mode.

The polymer end-product is supplied in a solid-grain form and is used in the manufacture of plastic products, *e.g.*, roof sheets, tanks, films and bottles. Depending on the type of application, the polymer grade that is used may differ in various chemical properties. The process considered here is capable of producing five grades (called grades A to E), each having further sub-grades. While all five grades use the same sequence of unit operations, the processing tasks used for each may vary in terms of the reaction conditions, separation

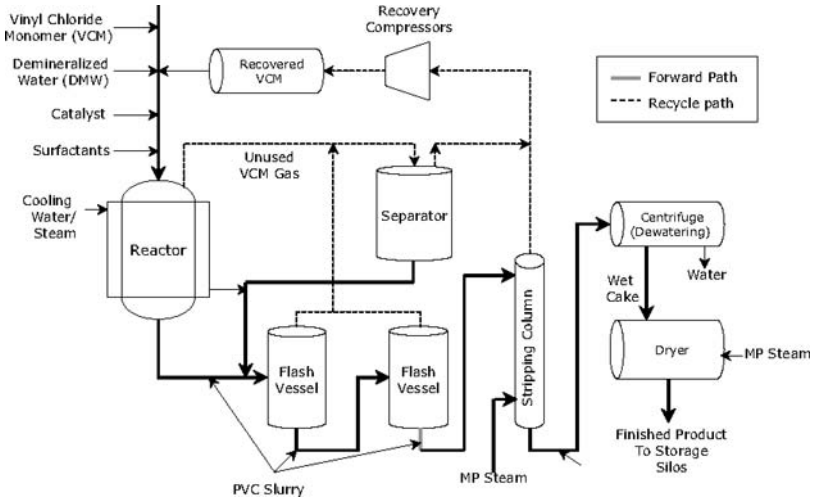

Fig. 4.8. Polymer process example

temperatures and the ratio and quality of recycled monomer to be allowed to be mixed with the fresh monomer. For instance, in a 'film' grade product only fresh monomer is allowed with stringent control of reaction conditions to achieve the desired quality of final product.

The process units consume various services for their processing tasks. The reaction occurs at a temperature between $50 - 90$°C, hence the feedstock entering the reactor is first heated to bring it to this temperature before it can enter the reactor. The reaction itself is exothermic and releases heat. This is removed via circulating cooling water (atmospheric temperature) and chilled water (4°C) in jacket and baffles of the reactor. The stripper columns use pressurised steam to purify the slurry and remove un-reacted monomer. The purified slurry thus contains extra heat which is conserved by heating the feed entering the reactor. The drier units similarly use pressurised steam to dry the purified slurry into a solid form.

Fig. 4.9 shows the layout of the process considered here. The layout offers a level of physical flexibility in terms of each line comprising certain parallel equipment that can be interconnected in various combinations. Each line can also be configured to produce a different polymer grade independently of the other line. Within each line, the reactors operate in a batch mode, hence individual reactor can be set up to produce a specific sub-grade without causing a significant mix up. The three stripper columns are shared between both lines such that only two columns are in operation while the third is being cleaned and regenerated. A similar facility exist to interconnect the centrifuge and drier units between lines, but this is not normally used unless necessary.

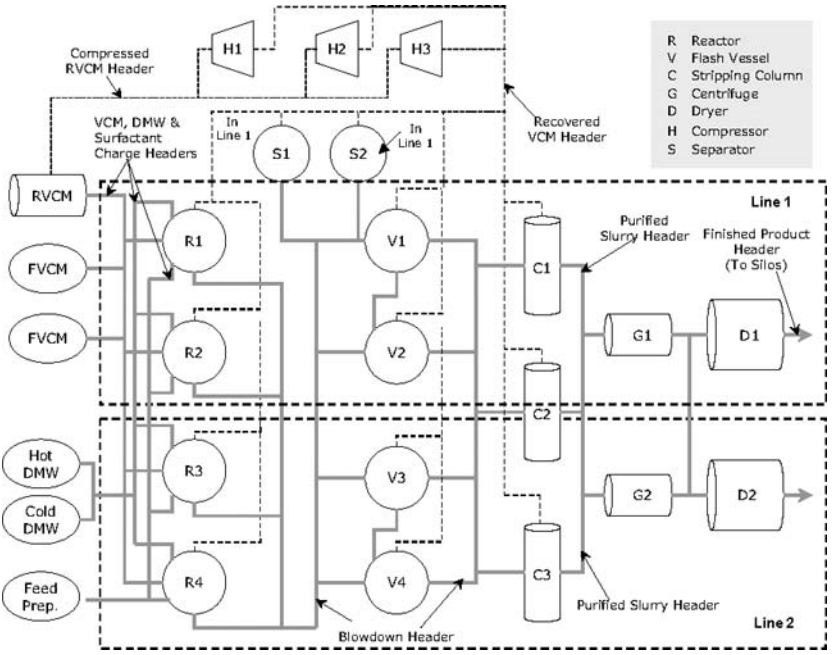

Fig. 4.9. Layout of the selected polymer plant

4.4.2 Identification of Process Elements

Based on the description in Section 4.2.1, the selection and assignment of process elements in the above process can be done as follows:

- *Unit Elements:* All process units, namely reactors, flash vessels, stripper columns, centrifuges, driers, and compressors are represented by individual unit elements. Each unit element possesses its local decisions that it regulates on its own. The reactor element, for instance, decides the yield of reaction (percentage of monomer converted to polymer) while the flash vessels and stripper columns decide the recycle flow of monomer. Note that these local decisions of unit elements interact physically due to their recycle connections. There often exists trade-offs. For example, the cost of separation and purification can be reduced if the reaction yield is increased, however this also means slow reaction times and hence reduced overall throughput. An optimum selection of local conditions is thus necessary to achieve global production goal.

- *Header Elements:* All 'switchable' piping networks in the process are represented by individual header elements. The identification is done based on materials or services being transferred, *e.g.*, monomer header, slurry header (or so-called blowdown header), purified product header, cooling

water header, *etc.* Although not shown in Fig. 4.9, the header elements also contain transfer equipments and heat exchangers where necessary.

- *Service Elements:* The supplier plants for all services (*e.g.*, cooling water, chilled water, pressurised steam) are represented by different service elements. An effective distribution of services remains crucial to plant operations. For example, the stripper columns must receive a minimum supply of steam at a pressure above certain value in order to produce an on-grade product. If the available supplies drop suddenly, then shedding of steam supply to other unit elements, *e.g.*, to driers, may be necessary with a simultaneous reduction in production throughput by unit elements.

- *Product Elements:* Each customer order for a separate sub-grade is represented by a product element. Multiple product elements may exist, although only two would be produced at a time as only two lines are available. The customer orders may span a number of campaigns, each campaign comprising multiple batches. A detailed schedule of recipe information, quality, quantity, *etc.* may be supplied as part of the definition of product elements but not all parameters may be needed depending on how the process of recipe management is managed between product and unit elements (see the next chapter for details).

4.5 Comments on the DRPC Architecture

Having proposed the new architecture and illustrated its use, we use the final section of this chapter to reflect on the features and properties of the architecture.

4.5.1 Comparison with Conventional Process Control

The proposed architecture differs from conventional approaches in that it is modular and distributed. The plant-wide control is decomposed into control modules of process elements. The network-level response of process operation emerges via direct interactions between these elements. To this end, it is first shown that the proposed architecture is compatible with conventional control systems. Apart from that, its distributed nature offers certain additional benefits in terms of the two main attributes: improved product/process diversity and modifiability, where it supersedes conventional architectures.

Compatibility with Conventional Control

The four process elements are sufficient to cover all control functions commonly performed by a conventional control system. In a fully distributed case,

the vertical hierarchy of control is decomposed into local control modules of elements. In an ideal case, the boundaries between planning, scheduling, optimisation and control levels are also blurred. The process elements solve the plantwide control problem at different level of abstraction depending on the nature of operating conditions and the disturbances arising, *i.e.,* operate in a long-term planning or proactive mode when there are no disturbances and in a short-term reactive mode when frequent disturbances are likely to arise. In the course of migrating to this fully distributed case, one can still consider an intermediate hierarchical form in which the elements first solve a planning problem in a distributed form and using its solution decide the set-points for lower-level scheduling or optimisation problems. By restructuring the control algorithms in this way one can develop and implement the same control functionality of a conventional system but now in a distributed way. This argument hence proves the sufficiency.

The functions of unit, header and service elements also relate directly to the physical decomposition of a continuous process into its constituent elements, *i.e.,* process units, piping networks and service suppliers. The product recipes of product elements specify the requirements of customer orders to bring together the physical elements to derive a process scheme. This is illustrated in Fig. 4.10. The presence of all four types of process elements is thus essential in a typical medium-to-large size plant to produce an end-product. The four types are thus also necessary to cover all control functions of a conventional control system.

Improved Product and Process Diversity

By using the notion of a product element, the architecture separates the procedural aspects of equipment control from the technical aspects of product recipes. This separation is important since it allows for the modification of both aspects in run-time. For instance, in case of frequent disturbances, it might be more sensible to consider an alternative product recipe or processing scheme than to reschedule the entire operation. Moreover, as discussed in the next chapter in detail, this integration is delayed until the stage where the actual production of a specific order is required, hence the most recent status of conditions on the plant can be taken into account.

Although the dynamic integration of recipe information can equally be incorporated in a conventional approach, the distributed nature of the architecture provides a sensible framework to implement it for two reasons. Firstly, the procedural control of process units is distributed, hence the equipment control can be easily modified. Secondly, the actual integration of product recipe occurs in a bottom-up manner by localised assignment of tasks to unit elements. As a result, emerging changes or disturbances can be managed in a graceful manner compared to conventional systems where this requires rescheduling the parts of or the full operation.

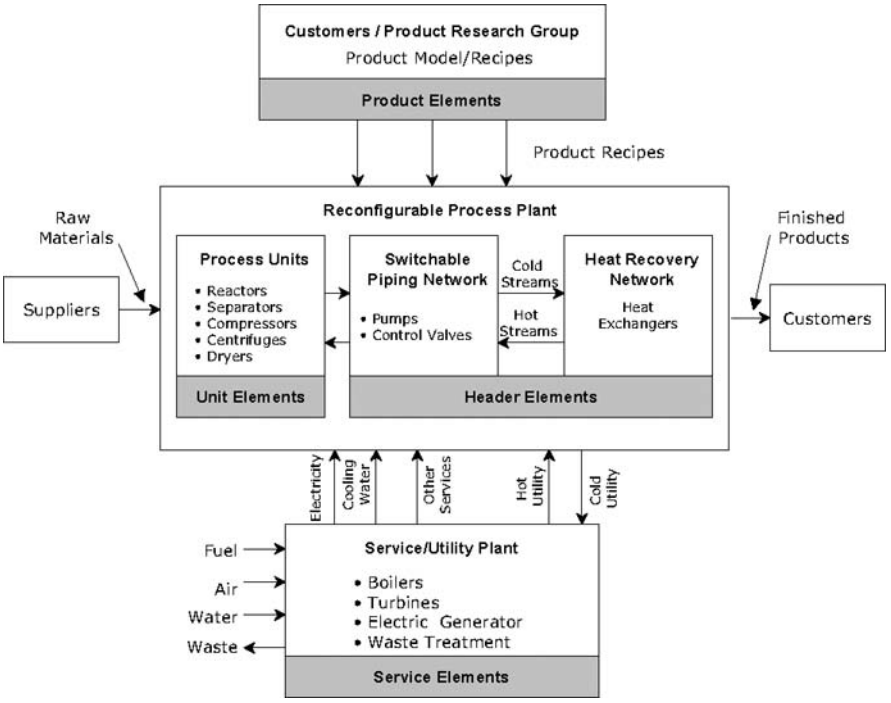

Fig. 4.10. Basic process elements cover all functions in a process plant

Easy Modifiability

The decoupling in local control of unit, header and service elements in the architecture can be expected to improve the modifiability of the control system. For example, one can easily replace a unit element with an equivalent another unit element provided their interfaces to other elements are same or compatible. The decoupling allows developing generic, multipurpose design of elements that can be standardised across a range of processes and re-used with little design and engineering effort as it has been the case for various package systems used in the industry, *e.g.*, industrial refrigeration.

Also, the use of header elements in the architecture introduces a further level of decoupling between unit elements. Unlike conventional control designs where each unit controller is pre-defined with exactly which other unit controllers it is connected with, in the proposed model the unit elements acquire this information on physical connectivity via header elements. The modifiability of the control system is hence enhanced in two respects: (a) it allows unit elements can be added or removed without changing the structure of their connections to other unit elements, *i.e.*, they only need to be defined with the header elements they are connected with, and (b) the level of flexibility supported in the design of header elements can be changed as necessary (*e.g.*,

by adding extra piping streams or transfer equipment) without changing the definitions of unit elements.

4.5.2 Comparison with Other Distributed Architectures

The proposed architecture retains the key features of previous distributed architectures in holonic or agent research (*e.g.*, PROSA (van Brussel *et al.* 1998) and its similar architectures). This is in terms of: (a) the product recipe information is kept separate from the procedures for equipment control, and (b) the architectural properties are kept independent of the control strategies of elements. The proposed architecture is thus equally able to manage the foreseen or unforeseen plant conditions as these other architectures. But, the proposed architecture also differs from the previous architectures in three respects as described below.

Introduction of Header Element Type

A new element type, the header element, is introduced to separate the control functions of transport mechanisms from that of processing tasks. A more advanced function than this could as well be assigned to header elements – that is to derive sequential operating procedures for the transfer of materials and services. Optimisation of the transport routes can equally be dealt with by header elements. Since these functions are implemented independently of the unit or service elements, a new form of decoupling is achieved that should help improve the modifiability of the control system.

Introduction of Service Element Type

A new element type of service element is also introduced to address the absence of a separate mechanism in other architectures to support the distribution of services. Although such functions are equally important in discrete manufacturing, they play an absolutely vital role in the timely and correct operations of process plants. Note that the service elements do not directly perform any processing tasks, hence could not, and should not, be represented by unit elements.

Specification of Interaction Behaviour of Process Elements

While the interaction behaviour of process elements is discussed at length in the next chapter, it suffices here to say that the role of product elements in the proposed architecture is different to that in the other architectures. The product elements interact with unit elements to map the product recipes onto production capabilities in the plant. Unlike PROSA or related architectures though, the product elements do not manage the logistics of materials or

services in the network nor do they define the operating conditions of these other elements. Such decisions are made by unit or header elements themselves once they are assigned with their tasks in the production.

4.6 Summary

In this chapter we have proposed a distributed architecture to support the reconfigurable process control. We next go onto examine how these elements in the architecture interact to allow the process to operate.