Multiscale Modelling as an Aid to Decision Making in the Dairy Industry

A thesis presented in fulfilment of the requirements of the degree of
Master of Engineering

Craig Alan Hutchinson
Department of Chemical and Process Engineering
University of Canterbury
Christchurch
New Zealand

March 2006

Abstract

This work presents the first known attempt to model the dairy business from a multiscale modelling perspective. The multiscale nature of the dairy industry is examined with emphasis on those key decision making and process scales involved in production. Decision making scales identified range from the investor level to the plant operator level, and encompass business, production, plant, and operational levels. The model considers scales from the production manager to the unit operation scale.

The cheese making process is used to demonstrate scale identification in the context of the important phenomena and other natural levels of scrutiny of interest to decision makers.

This work was a first step in the establishment of a multiscale system model capable of delivering information for process troubleshooting, scheduling, process and business optimization, and process control decision-making for the dairy industry. Here, only material transfer throughout a process, use of raw materials, and production of manufactured product is modelled. However, an implementation pathway for adding other models (such as the precipitation of milk protein which forms curd) to the system model is proposed.

The software implementation of the dairy industry multiscale model presented here tests the validity of the proposed:

- object model (object and collection classes) used to model unit operations and integrate them into a process,
- mechanisms for modelling material and energy streams,
- method to create simulations over variable time horizons.

The model was implemented using object oriented programming (OOP) methods in conjunction with technologies such as Visual Basic .NET and CAPE-OPEN. An OOP object model is presented which successfully enabled the construction of a

multiscale model of the cheese making process. Material content, unit operation, and raw milk supply models were integrated into the multiscale model. The model is capable of performing simulations over variable time horizons, from 1 second, to multiple years.

Mechanisms for modelling material streams, connecting unit operations, and controlling unit operation behaviour were implemented. Simple unit operations such as pumps and storage silos along with more complex unit operations, such as a cheese vat batch, were modelled.

Despite some simplifications to the model of the cheese making process, the simulations successfully reproduced the major features expected from the process and its constituent unit operations. Decision making information for process operators, plant managers, production managers, and the dairy business manager can be produced from the data generated.

The multiscale model can be made more sophisticated by extending the functionality of existing objects, and incorporating other scale partial models. However, increasing the number of reported variables by even a small number can quickly increase the data processing and storage demands of the model.

A unit operation’s operational state of existence at any point of time was proposed as a mechanism for integrating and recalculating lower scale partial models. This mechanism was successfully tested using a unit operation’s material content model and is presented here as a new concept in multiscale modelling.

The proposed modelling structure can be extended to include any number of partial models and any number of scales.

Acknowledgments

Thanks to Dr. Ken Morison for excellent supervision, sharing his dairy industry expertise, providing a sounding board for the development of ideas and suffering a continual bombardment of emails.

Henny Fairgray is acknowledged for relieving me of some of my domestic and social obligations.

Finally I thank Jim Anderton and my colleagues for giving me the support and flexibility at work to undertake this part-time degree.

Table of Contents

1 Introduction

Generally business modelling aims to provide the information needed by decision makers to maximise the profit making potential of the business and minimise exposure to risks and costs.

Because of the enormous number of inputs a complete business model could incorporate, and the varying levels of detail resolution required, business modelling evolved as individual stand alone models. Models tended to focus on the characteristics, and length and time scales of the system, relevant to the user. Typically only the most important variables were considered.

The result is often ad hoc system modelling, based on individual models used by different levels of decision makers. Individual models have minimal or no interaction between each other or the wider environment – often the only interaction between models is when a variable value generated by one model is manually input into a dependent model.

Multiscale modelling integrates individual models across wide time and length scales (from fractions of seconds to years; molecules to thousands of kilometres). Individual models simulate behaviour at different time and length scales. The models are connected (i.e. integrated), and data is transferred between them as and when required.

In a multiscale model, an input variable at one scale in the form of a constant, might be a dependent variable and an output variable of a model at a different scale.

1.1 Project Aim

The aim of this project is to:

- develop and test a feasible multiscale business model capable of supplying relevant decision-making information to decision makers from the plant to the boardroom in the dairy process industry.
- utilise developments in multiscale modelling theory, chemical engineering process modelling theory, and software development technology, to develop a multiscale model which would utilise a consistent data set.

Examples of these decision makers include plant operators, product production managers, marketing managers, utility production managers, and business managers. The types of information considered here includes scheduling, throughput, set points and production recipe details.

1.2 The New Zealand Dairy Industry

Table 1-1 shows global dairy industry production and export data for 2003. The New Zealand's dairy industry currently punches far above its weight in terms of its significance in the global dairy market. It is unique among major global producers in that the majority of production is exported. Among the four major dairy producing regions, New Zealand accounts for about 3% of global milk intake, yet accounts for 15% of global export market when trade within the European Union is included.

This position has been attained on the back of significant intervention in primary production by the New Zealand government from the mid twentieth century, followed by astute production and marketing diversification at the end of that century.

Farmers received guaranteed minimum prices, low interest loans encouraged people onto farms, and tax incentives encouraged investment and development. The industry benefited from research into animal husbandry and product development conducted in state funded research institutes and universities. International activities gave farmers cheap super-phosphate fertilizer resulting in rich pastures being maintained over decades. The New Zealand Dairy Board had a monopoly on all international marketing and sales activities involving dairy products.

Table 1-1 - Global Dairy Industry Production and Export Data (Source: Danish Dairy Board, 2003)

	Milk Production (million tonnes)	% Global Production	Exports (million tonnes)	% Global Export Market
NZ	13.9	2.8	1.73	15.1
Asia	113	23.1	0.28	2.4
Australia	10.6	2.2	0.75	6.5
European - EU	144	29.4	7.55	65.8
European - non EU	53.7	11	0.32	2.8
North America	99.3	20.3	0.38	3.3
South America	46.7	9.5	0.36	3.1
Middle East	9	1.8	0.1	0.9
	490.2	**100%**	**11.47**	**100%**

After experiencing near unrestricted access to its traditional markets for decades, changes in international trade rules in the 1960s and 1970s forced farmers and the Dairy Board to find new markets. Further change was brought on the industry in the 1980's and 1990's when subsidies and tariffs were removed, government research institutes were forced into market activities or privatised, and new international markets opened through the free trade agenda.

To compete internationally, merger and consolidation of the production side of the industry took place which has only recently ceased. Most recently, the Dairy Industry Restructuring Act, 2001, removed restrictions on the export of dairy products and promoted competition in the New Zealand domestic market.

Consequently, the modern New Zealand dairy industry is highly evolved and technologically advanced with a skilled knowledge based workforce. On-farm-management practice and animal healthcare are world leading. Production is characterised by large herd sizes and free range, grass eating stock. A diverse mix of commodity and niche products are manufactured at large, modern, multi-plant manufacturing sites which are distributed unevenly throughout the country.

The industry is dominated by farmer owned co-operatives, ranging from the massive Fonterra which manufacture a diverse range of products, to small niche product manufacturers. Fonterra deserve special attention.

The eventual result of a series of mergers between New Zealand's largest dairy co-operatives and the New Zealand Dairy Board, Fonterra is the world's 6th largest dairy company by sales (Rabobank International 2005 as referred on the Danish Dairy Board website). It accounted for about 97% of New Zealand milk production in 2003 and most of the country's dairy exports. Fonterra supplies the majority of fluid milk, cheese, and butter to the domestic market.

Fonterra owns processing facilities nationwide. In some dairying regions several manufacturing sites are clustered relatively closely together, providing management with production alternatives in normal and abnormal process situations. Other regions have only a few production sites several hours apart, limiting production alternatives, and severely limiting options in unplanned shutdown situations. Some sites process a single constituent material (e.g. lactose) and often separated constituent materials (e.g. milk fat) are transported to nearby processing facilities. Other sites have multiple processing options and process the complete milk product.

Though the New Zealand dairy industry has been a significant global player for many years, a feature of the industry is the low average price achieved for milk products compared with other global producers. The major contributing factor to this is the heavy weighting of production towards commodity products such as milk powder, cheese, and butter rather than high value or niche products.

2 Literature Review

To date there has been no known attempt to apply a multiscale modelling approach to aid decision making in the dairy industry. Therefore the reviewed material consisted of an examination of:

- existing process modelling tools
- existing multiscale modelling literature
- current and proposed mechanisms to facilitate the transfer of information between process models, specifically those which are developed for the transfer of information within a chemical engineering process modelling environment
- dairy process operations

2.1 Process Simulation Tools

Marquardt (1995) classifies process simulation tools into two groups, sequential modular and equation-oriented.

The modular approach allows the user to construct the process flowsheet from standardised modules, with each module modelling a unit operation (or part of it). The modules are linked to form the flowsheet. The module connections represent the material, energy and information streams of the process. The modular approach, though powerful and accessible to engineers for the solution of steady-state flowsheet simulation, does not adequately support the solution of more complex problems such as dynamic simulation.

In the equation-oriented approach, a set of equations which describe the system under consideration using balances for volume mass, energy, and momentum conservation, plus initial and boundary conditions, are constructed. These are then solved using mathematical techniques (section 2.2.1).

Equation-oriented models do not allow the construction of process models using engineering concepts such as the construction of a flowsheet from existing unit operation modules (i.e. models). This is readily achieved using sequential modular simulation. The historic inability to access a module model's equations resulted in inadequate levels of detail and minimal model reuse.

2.2 Multiscale Modelling

Multiscale modelling as a technique for improving the accuracy and efficiency of predictive modelling of engineering problems has been examined with increasing intensity since the mid 1990s (Cameron et al., 2005).

Charpentier (2003) describes the key factors driving this as:

- decreasing product development lifecycles (from 10 years in 1970 to 2 – 3 years in 2000).
- demands for less material and energy waste, near zero pollution requirements, defect free and safe products, and safe production.
- development of increasingly complex materials and compounds where control at the microscopic (and smaller) level is intrinsic to controlling the manufactured product quality.

Cameron et al. (2005) present a comprehensive coverage of the evolution and present state of research on engineering oriented multiscale process modelling and is used as the primary source on this topic. The definition, nature and characteristics of multiscale systems were identified in the context of the behaviour and rate processes which underpin the way scientists and engineers view the world.

The key issues associated with multiscale modelling, and strategies for their construction were discussed, principally:

- Selection of a specific modelling goal.
- Scale identification – an understanding of the time scales of interest often dictate the final scales of time and length needed.
- Model representation – in what form does the model exist and what forms are possible?
- Model integration – the linking of the partial models (i.e. those single scale models which make up the multiscale model) and the nature of information flow between partial models.
- Model solution – this is a huge area and remains a major challenge. Some aspects were briefly discussed.

A multiscale model is the composite model formed by the integration of partial models, where the partial models simulate important phenomena and other natural levels of scrutiny (Ennis and Lister, 1997) at different characteristic length, time or detail scales (Cameron et al., 2005).

Partial models are linked in some way (using a software solution) allowing the transfer of data between partial models. Much of the literature refers to the integration of partial models of different scales in a generic context as linking between a macroscale (large scale) model and microscale (small scale) model irrespective of the actual length or time scales under consideration.

2.2.1 Scale Identification and Model Representation

Both Charpentier (2003) and Cameron et al. (2005) define the generalized scales of interest, and discuss the different mathematical modelling techniques which are used to represent the different scales. Examples given are:

Scale	**Mathematical Modelling Technique**
Nanoscale	Molecular mechanics and computational chemistry.
Microscale	Molecular dynamics and computational chemistry.
Mesoscale	Computational fluid dynamics.
Macroscale	Dynamic simulation and process flowsheet simulation (sequential modular).
Megascale	Environmental and enterprise modelling.

There is now a large body of literature available describing attempts to develop multiscale models, and defining generalized characteristics, using a variety of applications.

For example, Freeden et al. (2004) attempted to improve the accuracy of ocean circulation modelling by integrating a global circulation partial model with a local circulation partial model. Quarteroni & Veneziani (2003) link a localized vascular flow perturbation partial model with a global blood circulation partial model in blood flow simulations to successfully predict the outcome of a surgical operation.

Much of the engineering literature covering this topic examines the interaction between the mesoscale (e.g. particle – particle, and particle – reactor wall interaction), the microscale (e.g. particle formation) and the nanoscale (e.g. reaction kinetics).

For example, Drews et al. (2005) present a multiscale model for simulating the deposition of copper onto computer chips by integrating a electrical resistance partial model at the microscale with a nanoscale partial model which simulates the electrochemical deposition of copper onto an initially flat copper surface. They also define a generic method for integrating multiple computer codes (representing partial models at different scales) and demonstrate its use.

Srolovitz et al. (1997) developed a model for diamond chemical vapour deposition across a range of length scales spanning 10 orders of magnitude. They integrated the microscale partial model, which provides the fundamental mechanism for diamond growth, with the mesoscale reactor geometry and operating parameter partial model.

Rey et al. (2004) developed a multiscale model which was able to provide fundamental principles for control and optimization of structures in polymer – liquid crystal material systems. This is an important chemical engineering application because 60% of all products sold by chemical companies today are crystalline, polymeric or amorphous solids (Charpentier, 2003). This work includes references to examples where multiscale material structure control at the nano- and macroscale is applied.

In a typical corporate business model, enterprise decision making is focused on a single goal – the creation of wealth for shareholders. Financial measures such as Shareholder Value Added (Ng, 2004), Net Present Value, and Internal Rate of Return (Ydstie, 2004) are used to gauge the actual or likely success of this wealth creation.

Ng (2004) presents a framework for linking financial measures to product and process design. For the reasons outlined earlier in this section, plus the recognition that the experience of scientists and engineers can often enhance the chances of success of a project, successful enterprise decision making will be increasingly integrated with the decisions of production and technical decision making. One response to this has been the development of the supply chain management and process simulation software tools now available.

All of the literature examined took an equation-oriented approach to model development. No literature was found which examined the dairy industry from a multiscale perspective.

2.2.2 Partial Model Integration

Cameron et al. (2005) examined the case for multiscale modelling. They discuss how it has been argued that multiscale modelling is simply the integration of existing software packages that model different scales. However they believe the evidence is that this approach may lead to lost conceptual opportunities, numerical inefficiencies and trouble shooting difficulties later on. Taking a more systematic approach to the definition and integration of partial models is an important feature of the implementation and performance of a multiscale model.

Steps have been taken towards understanding and classifying integration frameworks used in multiscale modelling studies.

Ingram et al. (2004) propose an integration classification scheme, consisting of five integration frameworks for linking partial models of different scales. The frameworks are divided into two broad classes:

- *Decoupled* frameworks (serial, simultaneous) are those where one partial model is solved, with the data generated by this model used as an input to its integrated model(s), which is in-turn solved.

- *Interactive* frameworks (embedded, multi-domain, parallel) involve the simultaneous solution of the constituent partial models.

The framework used in any particular integration depends on various properties of the partial models involved. e.g.:

- the portion of the system domain modelled,

- whether the models describe the same portions of the system domain at different levels of detail, or different adjoining parts of the system,

- the accuracy requirements,

- the direction of information flows between partial models,

- the purpose of the partial model(s).

A catalytic packed bed reactor case study was used to compare three of these integration frameworks linking a single catalyst pellet partial model (at the microscale) with a reactor bulk fluid phase partial model (at the mesoscale). The three frameworks tested produced similar (but not the same) predictions of system behaviour, but the integrated models displayed different implementation characteristics (i.e. effort for numerical solution, execution time, and memory requirements).

Some applications use data generated in a microscale partial model as an input into an adjacent macroscale partial model – data transfer is in a single direction. Other applications require bi-directional data transfer between adjacent microscale and macroscale models.

2.3 Process Modelling and CAPE-OPEN

The development of multiscale models in chemical engineering requires the integration of the software used to simulate (among others) production and unit operation partial models.

As discussed in section 1, process models have tended to focus on the particular characteristics of the system relevant to the decision maker who commissioned the model – resulting in ad hoc system modelling consisting of independent partial models having minimal interaction.

An important reason for this has been the physical inability of individual models to link to each other, let alone communicate and share information with each other. The CAPE-OPEN Project was the result of an attempt by the chemical and process engineering industry to develop standards for integrating individual process software models. It was a collaborative effort by a consortium of process industry, software industry and academic partners to define standard software interfaces.

Braunschweig et al. (2000) present an analysis of CAPE-OPEN and its application in the development and integration of unit operation modelling software components. The CAPE-OPEN documentation describes important concepts, and provides software specifications, for the construction of process modelling components capable of being integrated into a chemical process model.

Issues such as the construction of process flow diagrams and how to manage material stream, energy, and information flows between unit operations and process management applications are addressed.

The CAPE-OPEN Conceptual Design Document (2000) describes the conceptual ideas behind CAPE-OPEN, including:

- descriptions of the Process Modelling Environment (PME) and the various classes of Process Modelling Components – PMCs – (i.e. unit operations, numerical solvers, physical properties, and flowsheet analysis components).
- Material and material template, energy, and information streams, port and port type objects.
- connecting ports and sharing information between models.

2.3.1 Process Modelling Environment (PME)

The PME is the graphical user interface (GUI) for the creation of unit operations, flowsheet construction, and control of the simulation.

The PME is used to:

- define individual PMCs – here that involves creating individual unit operations based on a unit operation type template (which retains unique dimension and operational properties).
- construct a process flow diagram from individual unit operations.
- construct different production scenarios.
- manage the simulation and generated data.

2.3.2 Process Modelling Component (PMC)

PMCs are components that perform a specific function. Unit operations, numerical solvers, and physical and thermodynamic properties calculators are examples of PMCs. This work implemented unit operation PMCs.

2.3.3 Port and Port Type

A unit operation has a number of ports which allow it to be connected to other unit operations, and facilitate the exchange of material, energy, or information between other models.

A port has a given direction; inlet, outlet, or inlet/outlet. To facilitate the connection of like types of information, a port type is defined. The three types of Port are:

- Material
  A material port is used to facilitate physical material flows. They are the most complex to implement because of the amount of information needed to represent all the relevant properties of the material that might need to be used or calculated. Material ports represent the connection points for the material streams between unit operations.

- Energy
  An energy port is used to represent energy streams in the absence of material flows; for example, the heat loss from a unit operation or the transfer of energy through a motor shaft.

- Information
  Information ports are used to represent any other information flows which cannot be represented by either material or energy. An example might be where information from a downstream unit operation is used to set the flow ratio of the 2 outlets of an upstream unit operation.

2.3.4 Material and Material Template

A material port has a material object associated with it. The material contains all the data need to define the material (e.g. flowrate, temperature, constituent component information for a mixture). Between them, the material/port association allows the implementation of physical process streams.

The material is based on a material template. The material template provides the component and property information required, but not necessarily with values set. For example the milk material template has a collection of components (such as fat, lactose, protein, and water), plus physical and thermodynamic properties (such as density, viscosity, and specific heat capacity). Upon creation, the new material object inherits these components. Property values and component fractions can then be set.

It is not the intention here to produce a CAPE-OPEN compliant software model. However, because CAPE-OPEN presents a model for information sharing between different process modelling components such as unit operations (a key goal of this project), CAPE-OPEN concepts are implemented here. This also facilitates future CAPE-OPEN compliance.

The CAPE-OPEN standard is in the early stage of development. However, a reading of the CAPE-OPEN specification led to the conclusion that:

- the key material-energy-information / port concept which allows data to be shared via ports underpins CAPE-OPEN, so will not change significantly and,

- enough flexibility exists at the CAPE-OPEN interface to facilitate any minor changes that arise with either the port/information association or the connecting of ports, and

- the CAPE-OPEN specification has been designed to be independent of the type of process under consideration, therefore is applicable to the dairy industry.

The full CAPE-OPEN documentation is available in the internet at http://www.Co-Lan.org. It extends to detailed interface specifications (e.g. CAPE-OPEN Interface Specification - Unit Operations, 1999) for the construction of CAPE-OPEN compliant software models (both new and wrap-around for legacy code) developed in different software applications. Much of this material is targeted at software developers and is beyond the scope of this work.

2.4 Object Oriented Programming & VB .NET

The model developed here is constructed using the Microsoft Visual Basic .NET development environment, by creating object oriented programming (OOP) object classes.

Kurata (2001) gives a good overall discussion of OOP. In this work, the extensive and detailed MSDN Library online help (distributed with Visual Studio .NET 2003) is used as the sole reference for programming problems.

OOP is important because it is the software development technology which is used for building software components and applications today. The major benefits are it allows for efficient programming and efficient code reuse.

Once an object is built, much of the code can be hidden from the software developer (i.e. OOP encapsulation). This means the developer using the object does not need to know how the object performs its tasks. Only knowledge of what the object does, and what methods are used to achieve the task, is required.

For example if a developer uses a cheese vat object developed by a third party and wanted to fill the vat, a Fill function could be called. The Fill function might perform a series of operations to check that the cheese vat is not already full or that it is available for filling. If it can be filled the cheese vat object performs that task. If it can’t be filled the Fill function might return a false value indicating the fill operation was unsuccessful.

Another benefit of OOP is the efficient reuse of code. The creation of multiple instances of an object is very simple. Once the cheese vat template object is defined, a new instance of the cheese vat object can be easily created. Each new instance comes with the methods and properties of the original cheese vat template object.

One technology which facilitates the sharing of data between software components and applications is COM – (i.e. Microsoft’s Component Object Model). COM is a software architecture that allows applications and components that are built by different vendors to communicate, even when they are distributed among different computers and operating systems such as Windows, UNIX and Macintosh.

Microsoft's Visual Basic .NET is a software tool for the creation of software applications using COM objects and forms.

2.5 Dairy Processing

A dairy processing site consists of various unit operations and groups of unit operations, which taken together facilitate the production of the range of dairy products manufactured. Three processing stages are common to nearly all dairy processing facilities (Bylund, 2003).

1. Raw milk reception and storage.
2. Most countries require by law that some form of treatment be conducted on milk to destroy disease causing pathogenic micro-organisms.
3. Due of regulation or specification the milk will undergo fat content standardisation as an intermediate processing step.

The cheese making process as a series of unit operations is detailed in Jones (1999). Morison (1997) provides overall and unit operation mass balances at the constituent component level for a cheddar cheese making process.

The raw milk is received at the plant and stored. For efficient production the pasteurizer and separator must run continuously, so the milk treatment process is continuous. Raw milk is pasteurised, and excess cream separated from it before it is fed as standardised milk into the cheese vats. The cheese vat reaction process is a batch operation. Process continuity is maintained by filling the vats sequentially.

There is redundancy in the plant which allows the first vat to complete its batch cycle, be emptied and cleaned and in the fill queue ready to be filled again, before the final vat's first fill is completed. This also maintains a constant flow of curd onto the cheese belt, where most of the whey is separated out, before final processing in the block formers and the rapid cooling tunnels (both continuous processes).

3 A Multiscale Model of the Dairy Industry

At each decision making level of the dairy industry different decisions are made which will ultimately influence to a greater or lesser extent the immediate and future performance (in financial, social and environmental terms) of the plant, business and investment. Different information is required at different levels, covering different time scales, to facilitate good decision making.

There is little analysis of the relationships between the information required at various decision making levels, though clearly this information is integrated. Examples are given below.

A plant manager's ability to maintain and accurately forecast a plant's operation, and produce to-specification-product – thereby giving the greatest returns and minimal wastage – will influence:

- a production manager's decision to accept a supply contract of a particular duration and magnitude.
- a marketing manager's decision to pursue new markets.
- an investor's decision to invest (e.g. R&D, new processing).
- a regulatory managers obtaining of permits for pollutant discharges.

While a plant manager's ability to achieve this is influenced by:

- the production managers decisions (e.g. production schedules and product production sequence)
- the plant manager's own decisions (e.g. maintenance scheduling)
- decisions taken by the plant designer. For example, selection of plant items, measurement and control instrumentation, capacity, and plant

configuration will influence plant maintenance requirements, the ability to meet specification, and bottlenecks

- decisions taken by supervisors and operators such as cleaning or UF plant changeovers

Clearly the information transfer process between decision makers is complex and potentially endless. It is made more difficult because, while much information transfer occurs formally (in the form of reports based on data), many important decisions are made using informal information sources, such as experience or human networks. These are especially relevant when exception-event (e.g. unusual process conditions or unit operation failures) based decisions are made.

Multiscale modelling potentially has several applications to decision making in the dairy industry. For example:

- a major benefit will be a consistent data set for higher levels. With separate modelling, a forecast is produced at a low level (say a cheese plant production forecast), which is then passed onto a higher level to be used in a site forecast, and so on up the decision making chain. There is a time lag between the generation of each of these forecasts. It maybe that assumptions made at the lowest level is out of date by the time the data generated using those assumption is used at higher levels. The next point follows from this.
- providing faster, more accurate and detailed forecasting data.
- modelling the effect on processing facilities of changes to processing conditions (e.g. alternative products, flowrates, unit operation capacity).
- analysing the exposure of the dairy businesses profit to process scale variables such as production alternatives, and volatility in material costs and quality, and interest rates.

Limitations are placed on the degrees and scales of multiscale models by users (e.g. detail and accuracy requirements) and on users by data processing and software limitations.

3.1 The Multiscale Nature of the Dairy Industry

Consider an investor who owns among its many investments, a shareholding in a dairy manufacturing business which consists of multi-site and multi-plants. Each investment will have its own business model (partial models of the investor's model), including the dairy manufacturing business model. Each partial model in turn may consist of more than one partial models, depending on the accuracy and level of detail required.

3.1.1 Length and Time Scales

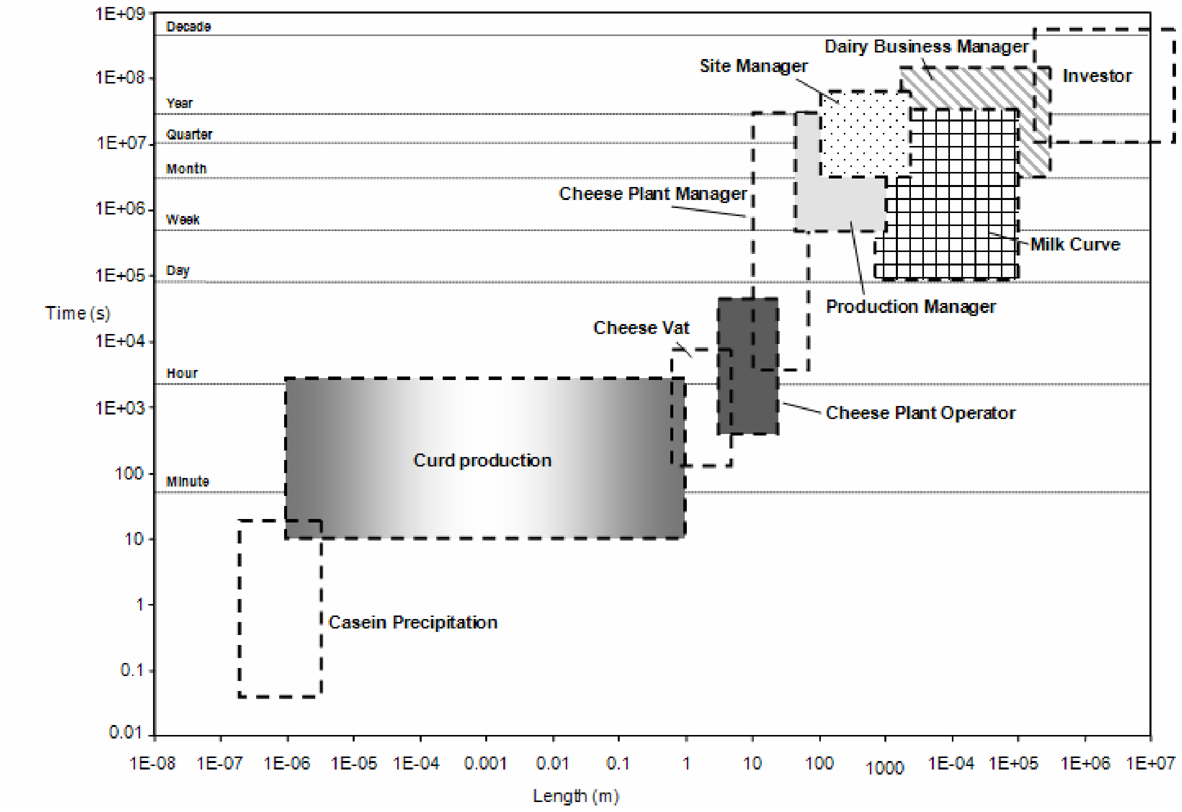


Figure 3-1- Scale Map for the Cheese Manufacturing Process

Figure 3-1 shows the scale map for decision making levels and control scales of a cheese making investment. The different components and their scales are discussed below.

3.1.1.1 <u>The Investor Scale</u>

An investor models the short term and long term profit of each business in the investment portfolio. The time scale of interest ranges from some minimum time, say 4 monthly reporting (the financial quarter), to the long term duration of the investment (possibly tens of years). High risk factors which increase the chance of failure of the investment may be reported on more frequently (e.g. the cash flow into new processing capacity during the construction and first months of operation). The characteristic length is the geographical spread of the investment.

3.1.1.2 Dairy Business Manager Scale

The dairy business manager (e.g. CEO) is responsible for carrying out the tasks (such as integrating acquisitions and achieving profit and growth targets) specified by the investor. The business manager is interested in the complete supply chain. Modelling is targeted at those factors affecting profit (such as sales, costs, and production) across the complete business environment.

Factors affecting production, sales, and distribution are of interest. The characteristic-time ranges between the minimum period of costs and sales monitoring (often monthly) and the maximum forecasting horizon. These will probably be aligned with the business plan (e.g. 5 years). As with the investor scale, the characteristic length could be global, regional, or domestic, based on the geographical distribution the dairy business.

The business manager receives many decision making inputs including one from the Costs level and another from the Marketing & Sales level. These scales lie between the business manager and the various cost and production centres of the business. All business units will feed data into one or both of these scales.

3.1.1.3 Site Manager Scale

The site manager ensures the smooth operation of a site, and will maintain an overview on all aspects of the dairy site’s operations. They have ultimate responsibility for the production, service, and administration operations of the site. A suitable characteristic time range could be from the daily summaries extending out to the end of the next production year.

Characteristic lengths are hundreds of metres up to kilometres depending on the geographic scale of the site. Those adjacent areas which affect the site's operation (e.g. natural resources as raw materials), and are affected by its operation (e.g. pollution), are considered.

3.1.1.4 Production Manager Scale

The production manager's responsibility is to deliver product on specification, on time, and on budget. Key areas of interest include the raw material supply, production, regulatory compliance, and manufacturing costs. They provide annual production data, costs, and product data (e.g. product specification) from each production facility.

Characteristic times of interest range from daily production to annual (and beyond) forecasts. Characteristic lengths are from tens to thousands of metres representing the physical distribution of production facilities.

3.1.1.5 Cheese Plant Manager Scale

The cheese plant manager is responsible for ensuring that cheese production schedules can be met. Phenomena and processes of interest include fouling rates for separation and heat transfer unit operations, cleaning regimes, reactor production (e.g. the cooking process in a cheese vat), raw milk supply, maintenance, the operation of individual unit operations, availability of services, and quality control are important.

Characteristic times of interest range from a few hours (i.e. the current production) to a month (the next month's production schedule). The characteristic length is the physical boundary of the cheese plant's processing operations.

3.1.1.6 Cheese Plant Operator Scale

The cheese plant operator's responsibility is to make the cheese. Factors which influence the operator's ability to produce on-specification product are important (e.g. processing conditions, the recipe). Characteristic times for this level could range from a few minutes (such as deciding the best moment to cut the curd in a cheese cooking vat) extending out to the end of the production day.

3.1.1.7 Cheese Vat Scale

The phenomena of interest are the curd production and whey expulsion processes that occur in a complete batch cycle. The steps involved in a cheese vat batch are (Bylund, 2003):

1. The vat is filled with pasteurised milk.
2. Starter bacteria and coagulating rennet is added.
3. The mixture is stirred to ensure uniform bacteria distribution.
4. The mixture is left for a period to coagulate into a solid curd gel.
5. The curd gel is cut into particles (curd grains) of the desired size. Whey expulsion from the curd gel begins.
6. The curds and whey mixture is then subject to a heating and stirring regime, which may include the removal of some whey and addition of hot water. This, combined with the bacteria growth that occurs, results in further whey being expelled from the curd grains.
7. After a final stirring period (the duration being determined by the desired pH and moisture content of the curd) the curds and whey phases are separated and removed from the cheese vat for further processing.

The characteristic length is an important dimension of the cheese vat (in the order of a few metres) while the characteristic time is the duration of a batch cycle.

3.1.1.8 Curd Production

The enzyme action of the rennet causes the casein in milk to precipitate and coagulate into a solid gel. Factors such as rennet type and distribution, temperature, pH, and calcium content of the milk govern this process.

Characteristic times are in the order of 20 – 30 minutes (the period covering the coagulation set stage of the cheese vat batch), while characteristic lengths are an important dimension of the cheese vat (of the order of less than a metre).

3.1.1.9 Casein Precipitation

The dominant type of protein in milk is casein (Bylund, 2003). Casein occurs in the form of micelles caused by the aggregation of sub-micelles. Each sub-micelle consists of a core of insoluble α_s-casein and β-casein, and κ-casein. Sub-micelles on the surface of the micelle have more κ-casein molecules in the surface that those sub-micelles on the interior of the micelle. This results in the casein micelle being hydrophilic, preventing its precipitation in the milk.

Curd is formed by the precipitation of casein micelles which occurs when chymosin enzyme cleaves some of the κ-casein. This allows the micelles to physically interact and form aggregates that precipitate.

Characteristic lengths between 5-10 nm (length of the κ-casein molecule) and 0.4 μm (diameter of a large casein micelles) are considered. Characteristic times are in the order of up to a few minutes (for the precipitation mechanism to occur).

3.1.2 The Milk Curve

The New Zealand seasonal milk production follows a predictable model known as the Milk Curve (Figure 3-2 shows Fonterra’s milk curve for the 2002/2003 and 2003/2004 production seasons, and is used for this discussion). The milk curve is a partial model of the production partial model. The milk supply goes from 2 to 3 million litres a day in the off season, increasing to a peak flow of over 65 million litres a day for up to a 12 week period. Milk flow then decreases in a linear fashion to the off season flow.

The organic and bacterial nature of milk means some processing (most importantly anti-pathogen treatment) must be conducted more or less immediately.

The enormous variation in milk supply throughout the production season has implications for product unit costs because plant capacity is under-utilised for much of the season, with at least 6 weeks complete shut down every year for many processing facilities.

The milk curve partial model has a time scale of 1 year. The length scale, representing the distances between farms and the processing facility, ranges from a few kilometres and a few hundred kilometres.

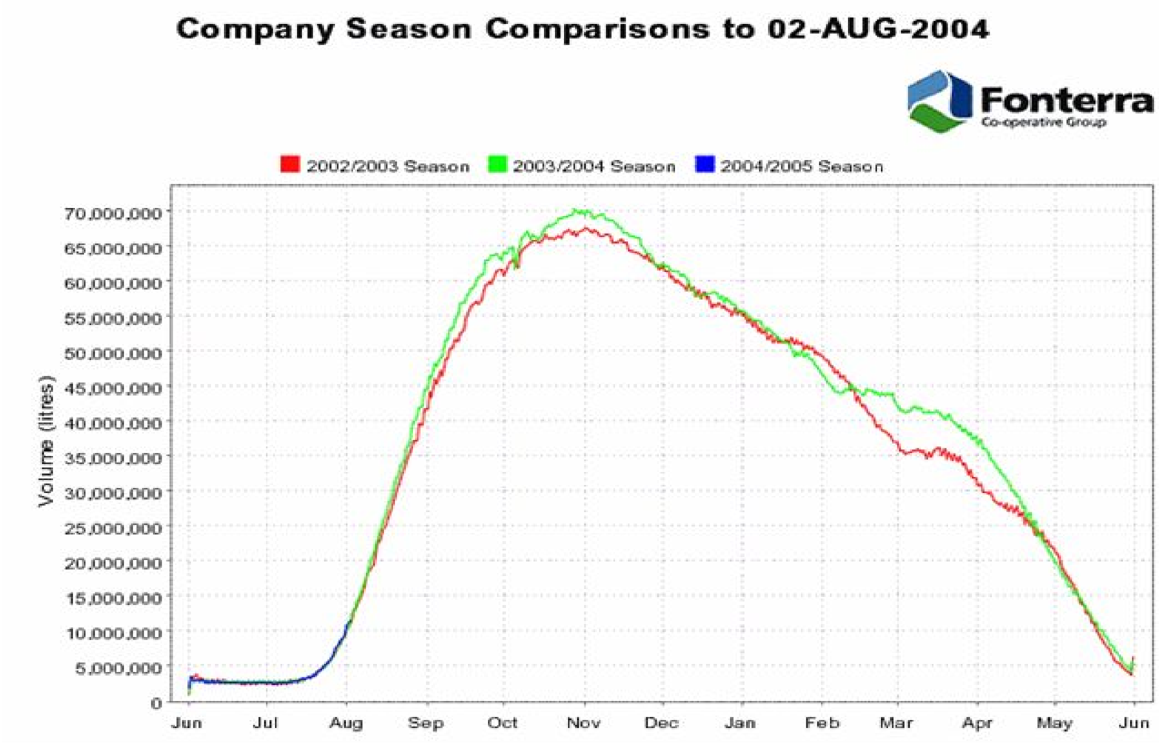


Figure 3-2 - New Zealand Milk Curve (Fonterra Personal Communication, 2004)

3.2 Dairy Industry Information Flows

Figure 3-3 shows the nature of some of the information flows in a cheese making investment at different scales, from the investor to the curd formation phase of a cheese vat batch. From a modelling perspective, information transfer occurs between adjacent scales.

Consider the investor's portfolio which consists of multiple investments. Each investment will contribute a changing profit or loss to the portfolio over time. The investor provides decision information (such as financial targets) to the dairy business, and in return receives financial information.

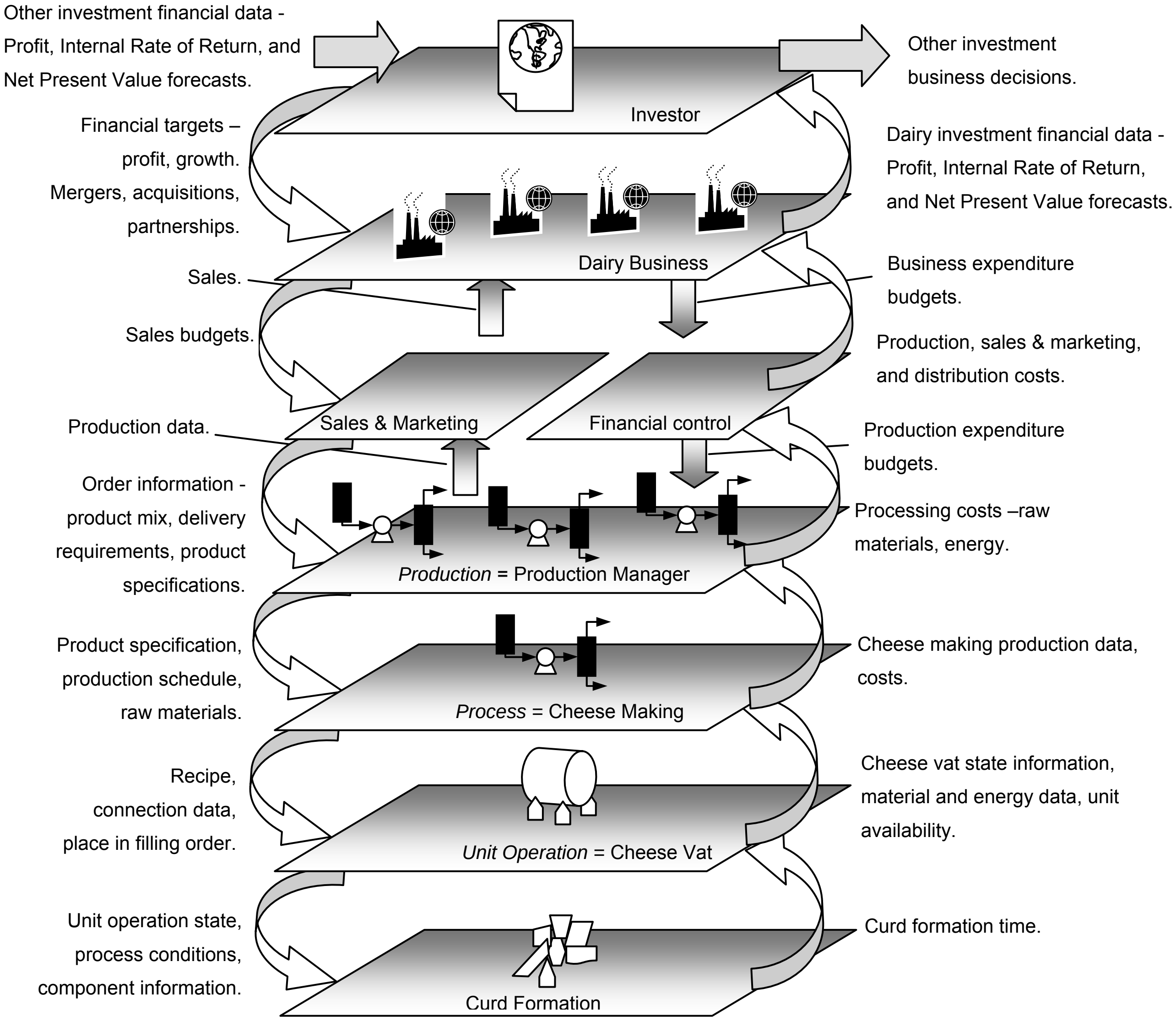

Other investment financial data - Profit, Internal Rate of Return, and Net Present Value forecasts.
Other investment business decisions.
Investor
Financial targets – profit, growth. Mergers, acquisitions, partnerships.
Dairy investment financial data - Profit, Internal Rate of Return, and Net Present Value forecasts.
Dairy Business
Sales.
Business expenditure budgets.
Sales budgets.
Production, sales & marketing, and distribution costs.
Sales & Marketing
Financial control
Production data.
Production expenditure budgets.
Order information - product mix, delivery requirements, product specifications.
Processing costs –raw materials, energy.
Production = Production Manager
Product specification, production schedule, raw materials.
Cheese making production data, costs.
Process = Cheese Making
Recipe, connection data, place in filling order.
Cheese vat state information, material and energy data, unit availability.
Unit Operation = Cheese Vat
Unit operation state, process conditions, component information.
Curd formation time.
Curd Formation

Figure 3-3 - A multiscale view of cheese production information flows

The dairy business scale takes the investor's financial targets and uses them to set sales and expenditure budgets, which are then used to define production requirements. These include product mix and delivery timing, along with expenditure budgets. Actual costs and sales are returned to the dairy business scale by the production scale.

The production scale provides product specification, scheduling and raw material information to the cheese making scale. Upon manufacture of the product, actual production and cost information is returned.

The cheese making scale provides an individual unit operation with information on how to behave. For example, when to turn on and off, when to clean, how much material to take, set points, and what unit operations it is connected to. The unit operation provides the cheese making process with information on how it is actually behaving. For example, the amount of material it contains, whether it is available for use, what its limits are (e.g. capacity, flowrate).

Considering the cheese vat, the formation of curd during the setting phase of a batch will depend on the component mix of the milk, and the process conditions within the vat. These will determine the duration of the set phase which dictates when the next processing stage in a cheese vat batch, the cutting phase, can occur.

3.3 The Modelling Goal

The aim of the dairy business is to make a profit. Several factors contribute to profit:

- Production
  e.g. plant operating conditions, production costs, product quality and availability.

- Business activities
  e.g. income, costs, marketing and distribution, R&D.

- Market forces
  e.g. prices, exchange rates, market growth, competition.

- Corporate activities
  e.g. income and costs generated by new investments, tax credits, depreciation, capital and interest repayments, dividends.

Other factors include environmental, regulatory, and political activities.

A comprehensive multiscale model of the dairy industry investment should incorporate partial models for each of those factors whose contribution to the investment’s profit is determined to be significant. One of the most significant contributions to profit comes from the production activity, and is the focus of this work.

The goal here is to develop a multiscale model capable of modelling different cheese production scenarios extending over different time horizons (hours, days, months, years). The scenarios represent the changing supply of raw milk over time and the final product alternatives available to the manufacturer.

The aim is to generate production information relevant to operational and planning level decision makers. On the financial side, only operating cost and value data related to processing, such as raw material, energy, and manufactured products is considered (i.e. costs such as wages and capital expenditure are not).

Either profit maximization, plant safety or environmental impact oriented decision making information delivery could be considered as potential business-level modelling goals for the multiscale model. The development of the cheese production multiscale model is a step towards all of these goals. The plant safety aspect is not considered, though partial models which factor safety will probably be inherent to any detailed process model. Here, only the financial aspect is considered. The environmental approach is examined in section 8.8.

3.4 Data Requirements and Partial Model Identification

Examples of the types of information needed to construct a model capable of achieving the modelling goal are given for production, cheese making, cheese operator and unit operation levels. From this, the required partial models can be identified.

3.4.1 Data Requirements

Production level decision makers needs the following information to generate a production schedule:

- raw milk supply over the time period of consideration. This is the milk curve discussed in section 3.1.2.
- order information so raw milk can be allocated to the various manufacturing options (that is if there is more than one option available).
- processing capacity of manufacturing facilities (e.g. the cheese making plant) and their availability. This information is also used to allocate milk supply when more than one manufacturing alternative is available.
- the amount of final product manufactured and when it is delivered.

The cheese making plant level decision makers need the following information:

- the production schedule for cheese making.
- the raw milk supply.
- the availability of processing equipment at the unit operation level.
- unit operation specifications. For example the capacity of silos and cheese vats, and pump flowrates.
- reaction information. For example the process conditions and steps within a cheese vat to produce curd which meets the specification.
- unit operation mass balance information. For example the behaviour of a cream separator at different flowrates and different input milk component ratios.

The cheese plant operator requires:

- a recipe (for example instructions on when to turn a pump on and off, or when to clean a unit operation).

- information on the process conditions in the cheese making processes unit operations (such as when the temperature within a cheese vat reaches the set point).

3.4.2 Partial Model Identification

From the data needs it can be seen that several partial models are required to create a multiscale model capable of generating the production information. These are:

- models of each unit operation which constitute the cheese making process.
- a model capable of calculating the addition and removal of material in a unit operation.
- a model of a cheese making process capable of generating production data. This includes unit operation connection information.
- a raw milk supply model because the raw material supply varies significantly over the annual milk production cycle.
- a production model capable of using the raw milk model and generating cheese production data from the process model over the desired time horizon.

4 Model Implementation

The model implementation required a process modelling environment (PME) and process modelling components (PMCs) for each of the unit operations in the cheese making process.

Two alternatives were available:

- purchase an existing PME from a vendor (such as AspenTech) and customize it to fit the dairy manufacturing processes being modelled, or
- construct a new PME and PMCs.

It was decided to construct a new PME and PMCs to model dairy industry processes for the following reasons.

- none of the existing PME vendors have developed modelling software for the dairy industry.
- the unique features of the dairy industry (e.g. the combination of the widely varying raw milk supply, the short lifespan of raw milk, the multiple production alternatives, the unique behaviour of reactors such as cheese vats, the unique properties of milk and its products, and the unique business rules such as processing equipment hygiene requirements).
- the author had access to the software code and could customize at the lowest level, rather than at the level dictated by the software vendor.
- the author had complete flexibility in accessing and formatting the data generated by the model.

The model implemented here is in the form of a sequential modular simulator modelling moving steady state behaviour. In other words, the simulator models the steady state behaviour of the process moving through time. It does not model the start-up behaviour of the plant, or the behaviour as the plant moves from cleaning to process fluid. For example, after cleaning, pipe work will contain cleaning water, which is purged by the first through milk. Milk will only be added to (say) the cheese vats when this water is removed and a pure milk stream exits the cheese vat feed pipe work.

Therefore, software solutions were required to:

- create unit operation models
- connect unit operations into a process flowsheet
- create production simulations for multi-product, multi-process, production scenarios over wide time frames
- run simulations of processes and production scenarios
- generate unit operation and process data

4.1 Implementation Software and Hardware

Software implementations were developed in Microsoft Visual Basic .NET. Microsoft Access 2003 was used for data storage.

The software design is based on object oriented programming (OOP) techniques. OOP has the major benefit that it allows the efficient reuse of code. CAPE-OPEN ideas are used to develop the object model structure for process and unit operation model development (see section 2.3).

The object model consists of Object and Collection classes. Their software implementation is an extension of the basic structure (i.e. methods and properties such as Add, Count, and Item described in any standard Visual Basic programming text – though none were referenced here). See Appendix D and E for samples of the object class and collection class code (i.e. Port and Ports) implemented here.

Simulations were performed using a HP Compaq nx7010 laptop running Windows XP, with a 1600MHz Intel Pentium M processor.

4.2 Object Model

An object model for the creation of unit operations, their connection into a process flowsheet model, and the creation and simulation of production scenarios is proposed. Key object relationships and hierarchy are derived from CAPE-OPEN.

The singular/plural convention denotes an object/collection respectively.

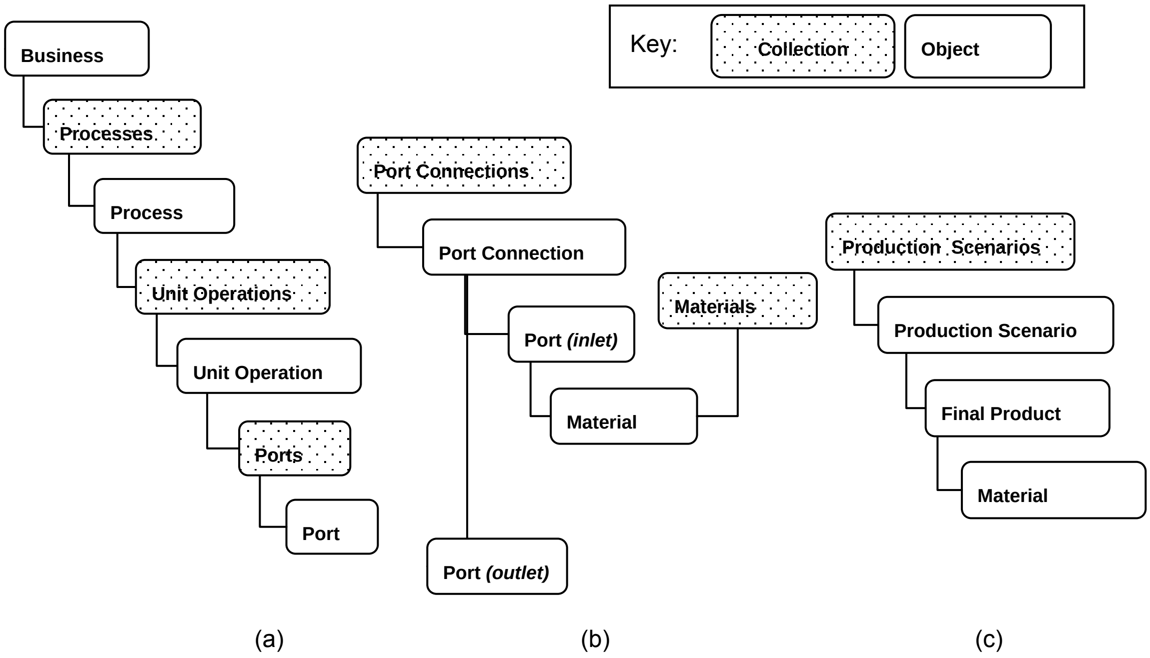


Figure 4-1 – Partial object model for a chemical process

Figure 4-1(a) shows the object hierarchy for a business as it relates to the production side of the business. A business consists of a collection of processes. Each process consists of a collection of unit operations. A unit operation's collection of ports facilitate flowsheet construction and are used in the transfer of material, energy, and other information between unit operations.

Figure 4-1(b) shows the relationship between a material port, its material, and its connections to other ports (the same principle applies to energy and information ports). A collection of Port Connections provides flowsheet connection information. Each port connection object defines the connection between two ports.

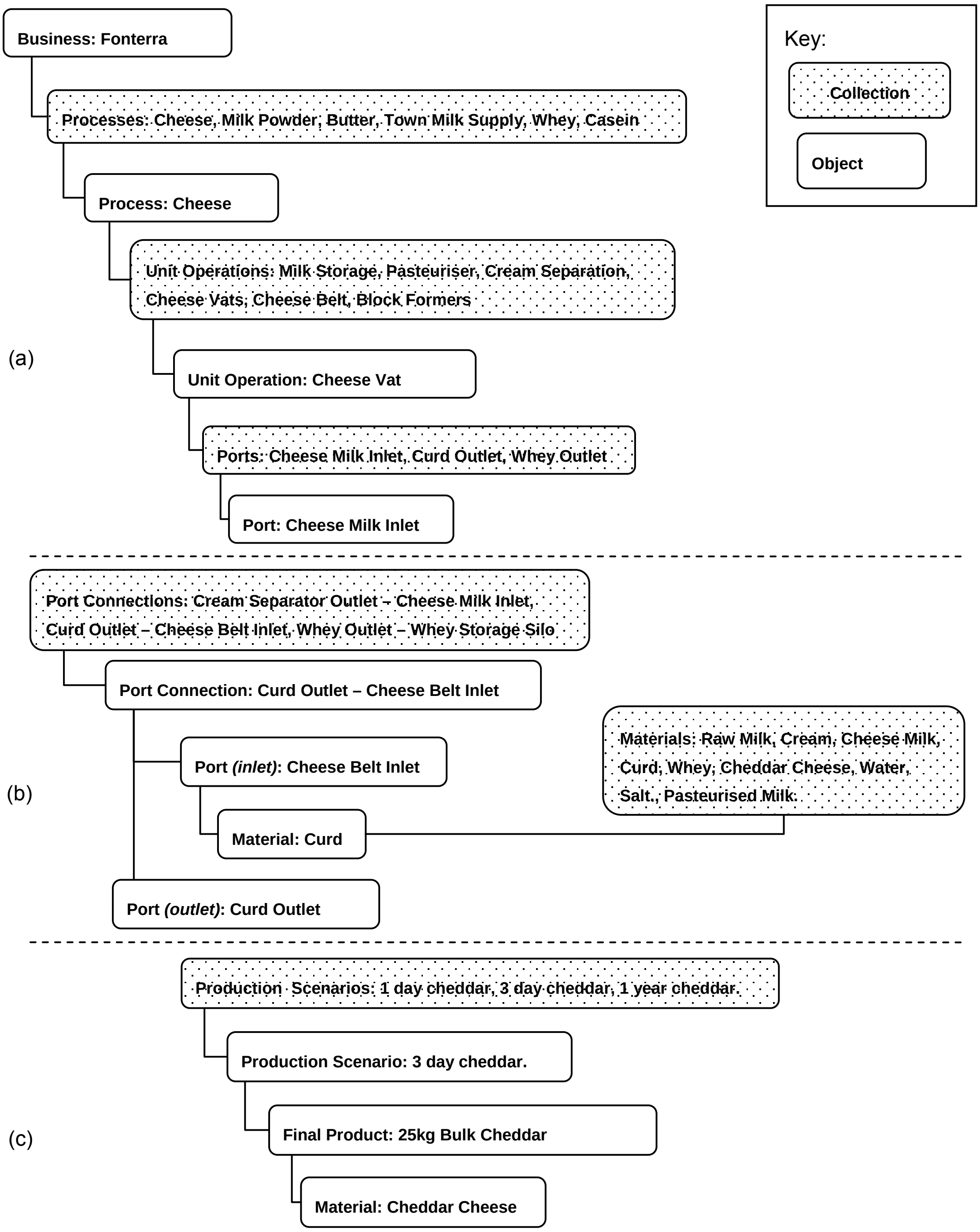

Key:
Collection
Object
Business: Fonterra
Processes: Cheese, Milk Powder, Butter, Town Milk Supply, Whey, Casein
Process: Cheese
Unit Operations: Milk Storage, Pasteuriser, Cream Separation, Cheese Vats, Cheese Belt, Block Formers
(a)
Unit Operation: Cheese Vat
Ports: Cheese Milk Inlet, Curd Outlet, Whey Outlet
Port: Cheese Milk Inlet
Port Connections: Cream Separator Outlet – Cheese Milk Inlet, Curd Outlet – Cheese Belt Inlet, Whey Outlet – Whey Storage Silo
Port Connection: Curd Outlet – Cheese Belt Inlet
Port (inlet): Cheese Belt Inlet
Materials: Raw Milk, Cream, Cheese Milk, Curd, Whey, Cheddar Cheese, Water, Salt., Pasteurised Milk.
(b)
Material: Curd
Port (outlet): Curd Outlet
Production Scenarios: 1 day cheddar, 3 day cheddar, 1 year cheddar.
Production Scenario: 3 day cheddar.
Final Product: 25kg Bulk Cheddar
(c)
Material: Cheddar Cheese

Figure 4-2 – Partial object model for a cheese vat in a cheese making process

Figure 4-1(c) shows the relationship between a production scenario which manufactures a particular product and, through the material object, a process that will manufacture that product.

Applying Figure 4-1(a) to a dairy business, Figure 4-2 shows the object hierarchy used to model a cheese vat in the cheese making process (the collections do not list all the possible constituent objects).

4.3 The Cheese Production Model

The cheese production model is created from:

- a unit operation material content (i.e. mass or volume) model
- the unit operation models (e.g. the cheese vat)
- a raw material model (e.g. the milk curve)
- the process flowsheet model (e.g. the cheese making process)
- unit operation state control (i.e. a modelling scenario)

4.3.1 Unit Operation Material Content Model

The material content within a capacitive unit (defined in section 4.4.1) is simulated using Euler’s method. The implementation of this is discussed in section 7.7.2.

4.3.2 Cheese Vat Unit Operation Partial Model

The classification and behaviour of unit operation models is discussed in section 4.4. Here, a description of how a unit operation model is defined is discussed. A cheese vat is used to demonstrate.

A cheese vat’s behaviour is controlled using its state property, with maximum volume as a boundary condition. The state of the cheese vat is defined as the current state of existence of the vat. At any point in time the cheese vat will exist in a particular state, and behave according to the rules for that state (e.g. when in the FILLING or EMPTYING state the cheese vat will recalculate its volume). A vat will have a collection of possible states.

Jones (1999) gives a description of a cheese vat batch process cycle. The exact specification for the batch depends on the type of cheese being produced. The description here is a general recipe for cheddar. In a cheese vat batch cycle the vat undergoes several state changes. These are:

- FILLING – milk and starter are added and the solution stirred to ensure an even distribution of starter. Enzyme in the form of microbial rennet is added; the solution is stirred again.
- SET – the solution is left to coagulate into a gel.
- CUT – once the gel is strong enough the gel is cut into a curds and whey suspension, followed by stirring.
- COOK – the vat temperature is ramped to the cooking temperature (37^{o}C) and the solution then cooked.
- STIRRING – the curds and whey is stirred until the desired pH is reached.
- EMPTYING – the vat is emptied onto the cheese belt.
- RINSE – the vat is rinsed with water and rejoins the FILL QUEUE.

The cheese vat can be placed in other states:

> FILL QUEUE and EMPTY QUEUE – states that indicate the vat is available for the transfer of material.
>
> CLEANING – a chemical clean will occur during a production run if that run continues for longer than some pre-defined period.
>
> OFF LINE – used when the cheese vat is not available to be used in the simulation.

The default state sequence for a cheese vat batch in this work is:

> FILL QUEUE > FILLING (until the maximum material capacity is reached) > SET (expires 30 minutes) > COOKING (expires 40 minutes) > CUTTING (expires 10 minutes) > EMPTY QUEUE > EMPTYING (until the vat is empty of material) > RINSE (5 minutes) > FILL QUEUE

The duration of the states (that expire) above are purely for the purpose of demonstrating the state transition mechanism proposed in this work, and may not reflect the actual durations found in any particular cheese making process. These durations will differ depending on (among other things) the specification of the cheese being manufactured, and the specification of the cheese vat's feed milk.

Between them, these states constitute the cheese vat's *State Collection* (see section 5.1 for the implementation of the State Collection). Knowledge of a unit operation's state collection is sufficient to model the behaviour of the cheese vat (but not the chemical processes within the cheese vat – see section 7.4 for a discussion on incorporating models of process reactions).

4.3.3 Raw Milk Partial Model

The raw milk model provides an amount of raw milk for a certain date. In this implementation the action taken is simply to look up a value in a data set. The process flowsheet model instigates this action by calling the raw milk model to supply an amount of raw material for a certain date.

In this implementation, an amount of raw material is made available to the raw material storage capacitive units on each 1 day iteration of the simulation. In the case of raw milk, daily supply data is obtained from the raw milk model, and stored in a raw materials collection – where it is available to all unit operations which require raw milk (e.g. for filling raw milk silos).

When an amount of raw milk is used by the process, the amount used is removed from the daily total available to the simulation. If raw material remains unused by the process at the end of any day's iteration, it will be the first to be used the next time a process requires that raw material.

Some raw materials expire after a period of time (e.g. raw milk) and are no longer available for use by the process. If raw material expires after it has been added to the process it can no longer be used by that process. In the case of raw milk this models its deterioration caused by bacterial growth.

4.3.4 Cheese Making Process Partial Model

Using unit operation information from Jones (1999) and mass balance data (stream flowrates and components) from Morison (1997), a generic cheese making process was modelled (Figure 4-3). The flowsheet is constructed via the addition of material ports to a port connections collection.

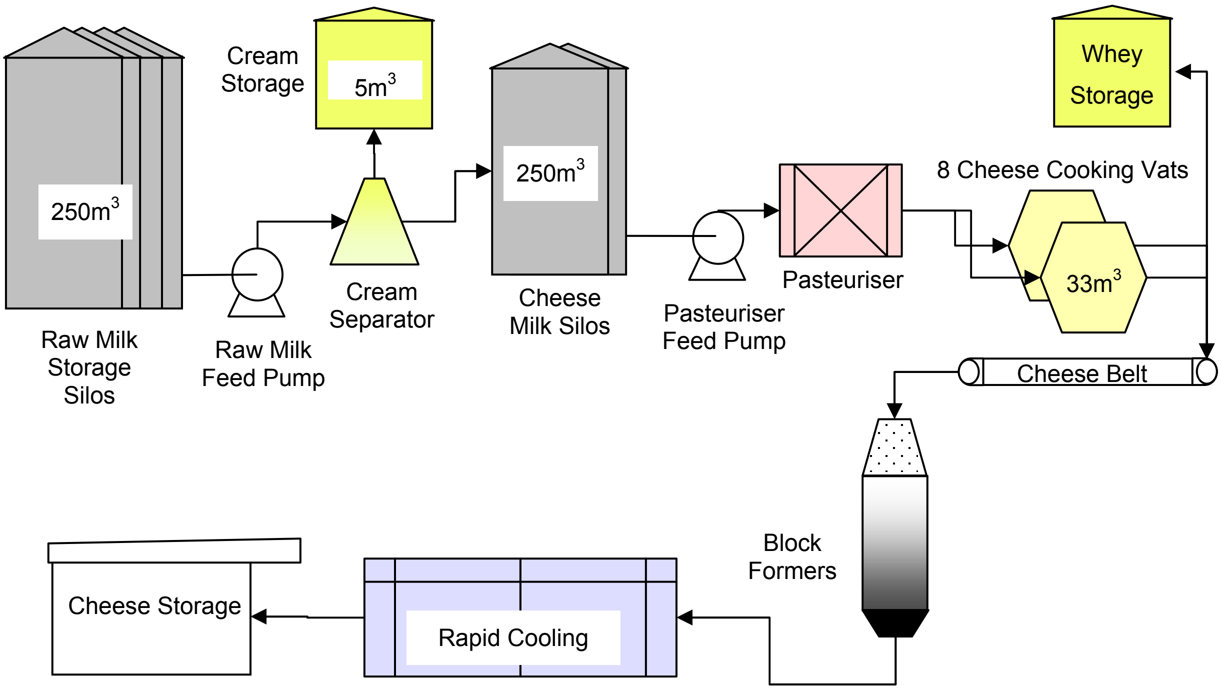


Figure 4-3 – Cheese Making Process (after Jones, 1999)

A port connection has two object properties (i.e. properties of the object in the object oriented programming context), an inlet port and outlet port. Inlet ports can only connect to outlet ports (or inlet/outlet ports, which aren't implemented here).

For example, a port connection with the cheese vat inlet occupying the inlet port property, and the cream separator skim milk outlet occupying the outlet port property represents the connection between the cheese vat and the cream separator.

The information required to transfer material between unit operations and control their behaviour is:

- The simulation's time increment (see section 4.10).
- the current state of unit operations, and their material port flowrates.

Once this information is known, a unit operation is sufficiently informed to recalculate itself.

For example, material transfer between unit operations is initiated by setting the state of a pump flow unit to ON. At this point, the pump checks that material is available to transfer (from an upstream capacitive unit), and there is somewhere to put the material (in a downstream capacitive unit). If both are available, the emptying and filling flowrates of the upstream and downstream capacitive units are set, and their states are changed to EMPTYING and FILLING respectively. The capacitive units, detecting these new states, recalculate the amount of material contained.

4.3.5 Unit Operation State Control

In order to facilitate the construction of a production model, two software classes are proposed:

- a Production Scenario class
- a Modelling Scenario class

4.3.5.1 Production Scenario Class

A Production Scenario class is defined as a set of controlling instructions which place unit operations in a desired state.

The Production Scenario is a 1 day interval, during which time the user controls the manufacture of product by passing instructions to unit operations. The selection of 1 day as the standard production modelling time period is arbitrary. However, 1 day is a useful period for integration with scheduling applications, and given raw milk supply arrives in 1 day batches.

The instructions change the state (hence the behaviour) of the instructed unit operation. For example, an instruction changing a pump's state from OFF to ON initiates an attempt to transfer material between unit operations on either side of the pump, while changing a cheese vat's state from OFF LINE to FILL QUEUE makes it available to receive material.

4.3.5.2 Modelling Scenario Class

A Modelling Scenario class is a collection of Production Scenarios which span the number of days required for the simulation. This is the mechanism by which a time scale is added to the production model. Production Scenarios are added to the Modelling Scenario in the sequential order with which the 1 day Production Scenarios are modelled. For example, a 30 day Cheddar Production Modelling Scenario will be constructed from 30 x 1 day Production Scenarios.

The process flowsheet model combined with the modelling scenario (and its constituent production scenarios) solve the production problem using a specified (and varying) supply of raw milk.

4.3.6 Model Simplifications

The model of the cheese making process was simplified. Several material streams are ignored. A more detailed and accurate model would include all the material streams, such as rennet and starter into a cheese cooking vat, the salt flow onto a cheese cheddaring belt, and centrifugal cream separation desludge. While some of these streams (such as rennet, starter and salt) are critical to the chemical reactions and consequently final product specification, they do not significantly affect the mass flows.

Reaction scale and mass balance partial models are not implemented. If they were these streams would need to be included in the model.

4.4 Unit Operation Classification

Two generic classifications of unit operation are implemented – a Capacitive Unit and a Flow Unit. Every unit operation in the cheese making process is one (or a combination of more than one) of these classes. These classifications are used for behaviour control – in this case hard coded within the capacitive unit and flow unit software classes.

4.4.1 Capacitive Unit

A capacitive unit is a unit operation which has material storage capacity. Two types of capacitive unit are identified:

1. Material storage unit operations. Their primary purpose is the batch storage of quantities of material (e.g. storage silos, cheese vats).
2. Unit operations which operate continuously but contain significant amounts of material within their modelling boundaries. (e.g. a block former or a cheese belt).

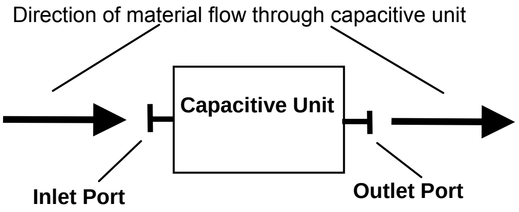


Figure 4-4 –Capacitive Unit

Here, a capacitive unit (Figure 4-4) has no more than one inlet and one outlet port. Although most capacitive units will have one of each, capacitive units which have no port in one direction are defined as either a raw material storage (and are filled from the Raw Materials collection) or final manufactured product storage (depending on which port direction is missing). Separation operations are implemented using Flow Units (section 4.4.2).

The Capacitive Unit class has code to perform the following:

- Define the unit operation as a raw material storage vessel (e.g. has no inlet material ports).

- Define the inlet material as a manufactured product, thereby defining itself as a product storage vessel.
- Indicate whether it is available to store or release material.
- Recalculate its volume when one or more of its material port's flowrates is non zero.
- Fill and empty simultaneously.
- Pre-check the next state and update to a new state.
- Generate unit operation operational data e.g. state, volume, material stream flowrates, energy flows.

Capacitive units as defined here have at least 5 potential default states:

OFF LINE, FILL QUEUE, FILLING, EMPTY QUEUE, EMPTYING

Other states are user defined, e.g. a cheese vat's states:

SET, COOKING, CUTTING, RINSE, and CLEANING

4.4.2 Flow Unit

A flow unit is probably easiest to define as a unit operation that isn't a capacitive unit (i.e. one that doesn't store material). Though all unit operations have some material capacity (e.g. a pump's impeller chamber or a heat exchanger's material space), here a flow unit has been defined as a unit operation:

- for which material storage is not a primary purpose;
- which has an insignificant capacity (in the context of the process);
- which has a low material residence time (a low residence time might be in the order of a few seconds to a few minutes);

Flow units are modelled as having no residence time. In this work they have two default states, OFF, and ON. Two types of flow unit are modelled:

1. A flow generating flow unit (Figure 4-5). This type of flow unit has a non-zero default flowrate. They have one inlet and one outlet port. Pumps and conveyers are examples of this type of flow unit.

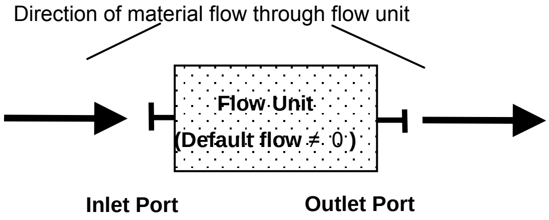


Figure 4-5 – Flow Generating Flow Unit

2. A flow unit which performs some processing operation on the material as it passes through the unit. This type they can have multiple inlet and outlet ports. Mixers, separators and heat exchangers are examples of this type of flow unit (Figure 4-6).

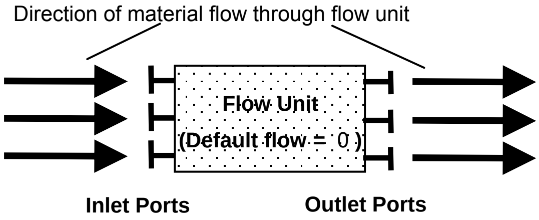


Figure 4-6 – Flow Unit

4.5 Capacitive Unit / Flow Unit Interaction

The cheese making process is made up of capacitive units and flow units connected together, each with a different role. As discussed in section 2.3.1, the Process Modelling Environment (PME) is used to manage the inter-connection of unit operations and construct the process flow sheet. The implementation of this is discussed in section 5.3.

Capacitive units store material and provide a unit operation for a reaction. Flow units drive the transfer of material throughout the process, mix or separate material, or represent a non-capacitive unit operation (e.g. a heat exchanger). There are many actual connection configurations which exist in processes. For example flow units are connected to flow units (e.g. pumps feeding heat exchangers). Flow units are connected downstream of capacitive units (e.g. pumps drawing material from storage silos. Flow units are connected upstream of capacitive units (e.g. pumps feeding storage silos). Flow units connected to multiple capacitive units (e.g. heat exchanger connected to multiple cheese vats). These do not complete the possibilities.

The following connection regimes were implemented here to construct the cheese making process:

1. A flow generating flow unit is upstream of a capacitive unit (e.g. a milk tanker empting pump used to fill a raw milk storage silo – not implemented here), represented in Figure 4-7). The flow unit is used to fill the capacitive unit (the assumption is that upstream material is available). The flow unit generates flow (i.e. its default flowrate property value is non-zero).

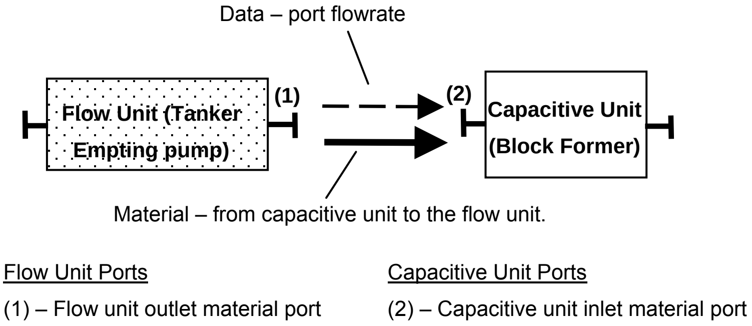

Data – port flowrate
Flow Unit (Tanker Empting pump)
(1)
(2)
Capacitive Unit (Block Former)
Material – from capacitive unit to the flow unit.
Flow Unit Ports
(1) – Flow unit outlet material port
Capacitive Unit Ports
(2) – Capacitive unit inlet material port

Figure 4-7 – Flow Generating Flow Unit –Capacitive Unit Downstream

When the flow unit's state is changed from the OFF state to the ON state, the following steps occur:

   a. The process modelling environment (PME) checks that the downstream capacitive unit is available for filling (e.g. state: FILL QUEUE; current volume is less than maximum volume).

b. The flow unit's inlet port's flowrate (1) is set. The PME sets the flowrate of the capacitive unit's outlet port (2) which is connected to port 1.

c. The milk silo capacitive unit, detecting a flowrate at one of its ports, recalculates its volume. Here data is transferred (i.e. the flowrate) in the same direction as material flow.

2. A flow generating flow unit is downstream of a capacitive unit (e.g. a pump connected downstream of a milk storage silo) – Figure 4-8. The flow unit is emptying the capacitive unit (the assumption is that a downstream capacitive unit is available for filling). The flow unit generates flow.

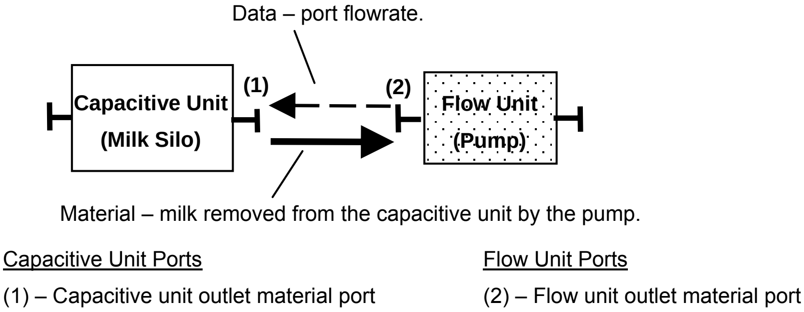

Data – port flowrate.
Capacitive Unit
(Milk Silo)
(1)
(2)
Flow Unit
(Pump)
Material – milk removed from the capacitive unit by the pump.
Capacitive Unit Ports
(1) – Capacitive unit outlet material port
Flow Unit Ports
(2) – Flow unit outlet material port

Figure 4-8 – Flow Generating Flow Unit – Capacitive Unit Upstream

When the flow unit's state is changed from OFF to ON, the following steps occur:

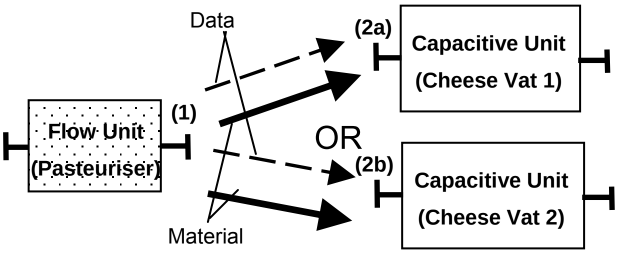
a. The PME checks that the capacitive unit is available for emptying (e.g. state: EMPTY QUEUE; volume is non zero).

b. The flow unit's inlet port's flowrate (2) is set.

c. The PME sets the flowrate of the capacitive unit's connected outlet port (1) which is connected to the flow unit.

d. The milk silo capacitive unit, detecting a flowrate at one of its ports, recalculates its volume. Here data is transferred (i.e. the flowrate) in the opposite direction to the material flow.

3. A flow unit is connected to, and filling, more than one upstream capacitive unit (Figure 4-9). Capacitive units are filled sequentially. For example, multiple cheese vats are placed in parallel, and filled sequentially. The aim is to operate the pasteurizer (at the cheese vat's inlet), and the cheese belt (at the cheese vat's outlet) continuously.



Flow Unit Ports
(1) – Flow unit outlet material port

Capacitive Unit Ports
(2) – Capacitive unit inlet material port

Figure 4-9 – Flow Unit – Sequential Capacitive Units Downstream

When the flow unit's state is changed from OFF to ON, the following steps occur:

a. The PME selects a downstream capacitive unit which is available for filling.

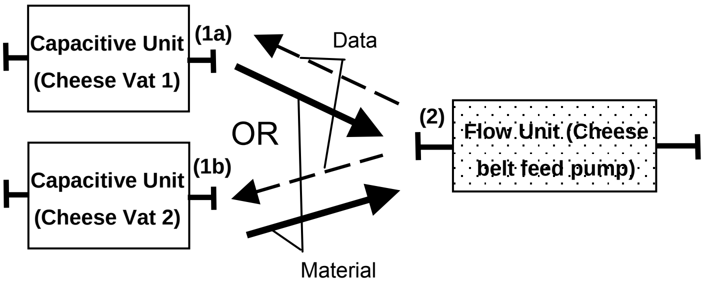
b. The flow unit's outlet port's flowrate (1) is set.

c. The PME sets the selected capacitive unit's connected inlet port's (2a) flowrate.

d. When the capacitive unit is full, the PME finds another downstream capacitive unit which is available for filling.

e. The PME sets the selected capacitive unit's connected inlet port's (2b) flowrate.

4. A flow generating flow unit is connected to more than one upstream capacitive unit (Figure 4-10). An example is the sequential emptying of cheese vats onto a cheese belt.


Capacitive Unit (Cheese Vat 1)
(1a)
Data
OR
(2)
Flow Unit (Cheese belt feed pump)
Capacitive Unit (Cheese Vat 2)
(1b)
Material

Capacitive Unit Ports

(1) – Capacitive unit outlet material port

Flow Unit Ports

(2) – Flow unit inlet material port

Figure 4-10 – Flow Generating Flow Unit – Sequential Capacitive Units Upstream

When the flow unit's state is changed from OFF to ON, the following steps occur:

a. The PME selects an upstream capacitive unit which is available for emptying.

b. The flow unit's inlet port's flowrate (2) is set.

c. The PME sets the selected capacitive unit's connected outlet port's (1a) flowrate.

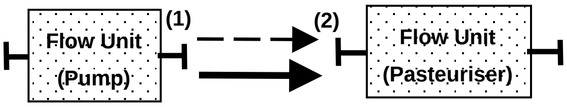
d. When the capacitive unit is empty, the process controller finds another upstream capacitive unit which is available for emptying.

e. The PME sets the selected capacitive unit's connected outlet port's (1b) flowrate.

5. Two flow units are connected in series (e.g. a pump connected to a pasteurizer). Here, the flow generating flow unit is connected upstream of a non-flow generating flow unit (Figure 4-11). When the flow generating flow unit's state is changed from OFF to ON, the following steps occur:

a. The flow generating flow unit's outlet port's flowrate (1) is set.

b. The PME sets the flow unit's connected inlet port's (2) flowrate.



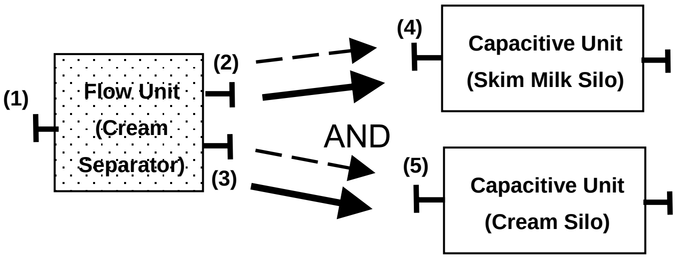
Flow Unit Ports

(1) – Capacitive unit outlet material port

(2) – Flow unit outlet material port

Figure 4-11 –Flow Generating Flow Unit – Flow Unit Downstream

6. A flow unit (both flow generating and non-flow generating instances were0020implemented) is connected to more than one upstream capacitive unit (Figure 4-12) through different outlet ports. Each flow unit port provides flow to its connected capacitive unit.



Flow Unit Ports	Capacitive Unit Ports
(1) – Flow unit inlet material port (raw milk)	(4) – Capacitive unit inlet material port (skim milk)
(2) – Flow unit outlet material port (skim milk)	(5) – Capacitive unit inlet material port (cream)
(3) – Flow unit outlet material port (cream)	

Figure 4-12 – Flow Unit – Multiple Capacitive Units Downstream

For example, a cream separator splits flow into two streams (i.e. skim milk and cream). One stream goes to the skim milk silo capacitive unit, the other to the cream silo capacitive unit. For the transfer to be successful, both capacitive units must be available for filling. The flow is split according to some pre-defined ratio, so the sum of the outlet port's (2 and 3) flow equals the inlet port (1) flow. Given each capacitive unit can be filled, flows at ports 4 and 5 are set to the flows at ports 2 and 3 respectively.

When the flow generating flow unit's state is changed from OFF to ON, the following steps occur:

a. The PME checks that the capacitive units connected to the flow unit outlet ports are both available for filling.

b. The flow unit's outlet ports flowrates (2 and 3) are set.

c. The PME sets the flow unit's connected inlet ports (4 and 5) flowrates.

4.6 Multi-class Unit Operations

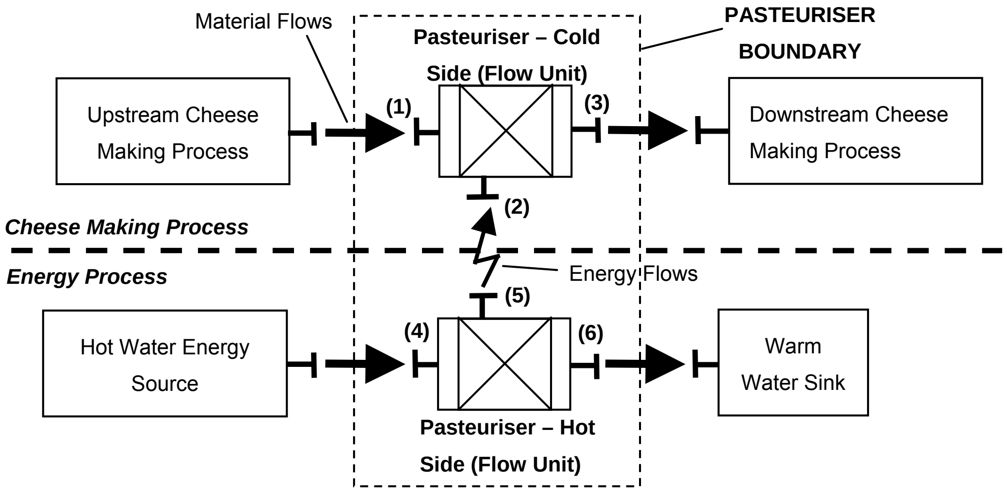
Some unit operations are modelled using a combination of unit operation classes.

4.6.1 Heat Exchanger

A heat exchanger is modelled using two flow units, a cold side flow unit and a hot side flow unit. In the case of a plate pasteurizer, the cold side (milk) is part of the cheese making process model. The hot side is part of a separate energy supply model.

In Figure 4-13 the pasteurizer cold side is connected to the cheese making process (ports 1 and 3). The hot side (superheated water) provides the energy needed to pasteurise the milk, and is connected to an energy source and energy sink (ports 4 and 6).

Energy ports 2 and 5 provide the connection and energy transfer mechanism between the cold side and the hot side. In the cold side the milk temperature increases, while in the hot side the hot water temperature decreases, as they pass through the flow units.



Pasteuriser Cold Side Ports

(1) – milk inlet (material port)

(2) – pasteuriser energy inlet (energy port)

(3) – pasteuriser milk outlet (material port)

Pasteuriser Hot Side Ports

(4) –Hot water inlet (material port)

(5) – Energy outlet (energy port)

(6) – Warm water outlet (material port)

Figure 4-13 –Information exchange in a multiple flow unit pasteuriser

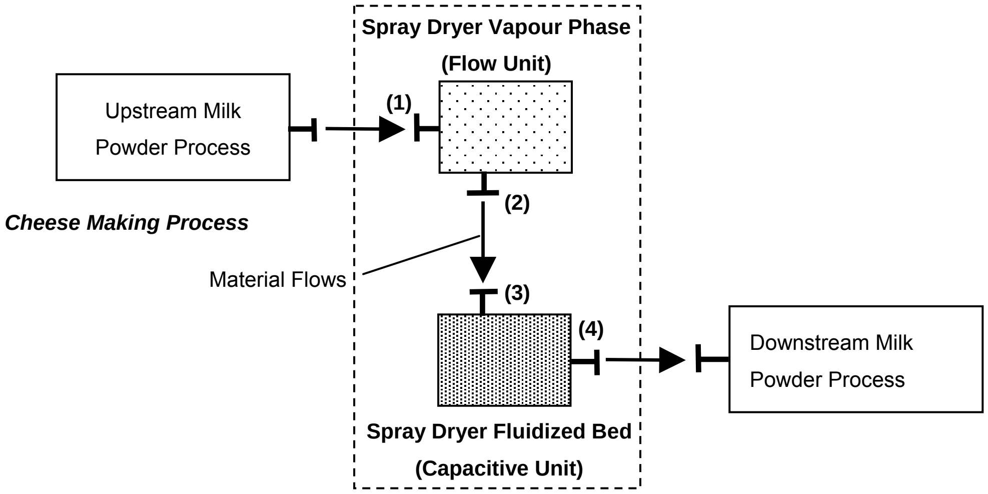
This situation was successfully implemented, though more work is needed on the Energy object to enable more sophisticated energy modelling.

4.6.2 Spray Dryer

Though not part of the cheese making process, another example of a unit operation important in the dairy industry which could be modelled using combinations of flow and capacitive units is a spray dryer. In order to examine the potential of multi-class unit operations the following configuration was implemented, and successfully tested

A spray dryer is modelled using a flow unit upstream of a capacitive unit (Figure 4-14). The flow unit represents the atomization of milk concentrate and the dryer vapour space, and also provides the flowrate for the transfer into the spray dryer. The capacitive unit represents the holdup of powder in the dryer and the fluidized milk powder bed at the bottom of a spray dryer. This models the accumulation of milk powder within the spray dryer unit.

Connection to the upstream and downstream processes are via ports 1 and 4. The vapour phase flow unit's outlet material port (2) is connected to the milk powder fluidized bed's inlet material port (3).



Vapour Phase Ports
(1) – Concentrated milk inlet (material port)
(2) – Milk powder outlet (material port)

Fluidized Bed Ports
(3) – Milk powder inlet (material port)
(4) – Milk powder outlet (material port)

Figure 4-14 –Information exchange in a flow unit / capacitive unit spray dryer

4.7 Modelling Continuous Flow in a Capacitive Unit

In the cheese making process, some unit operations perform both flow generation and material storage functions. This class of unit operation operates continuously, and essentially transfers material within the process. It is distinct from a standard flow unit (such as a pump or separator/mixer) in that it has a non-negligible material residence time.

Note that above, non-negligible is not defined. The decision to model a continuously operating unit operation’s capacity will depend on how significant that capacitive nature is to the operation of the process. For example if the unit operation holds up (say by more than a few minutes) the operation of the process downstream while it fills and material moves through it than it may be desirable to model capacitance. The cheese belt and the block forming tower are examples of this.

The cheese belt is a belt conveyer. The block forming tower is essentially a vertical sided storage silo which continuously expresses compressed curd at the bottom of the silo at the same flowrate as the curd is added to the top of the silo.

When not filling or emptying at the start and completion of a process run, they function as continuous plug-flow unit operations.

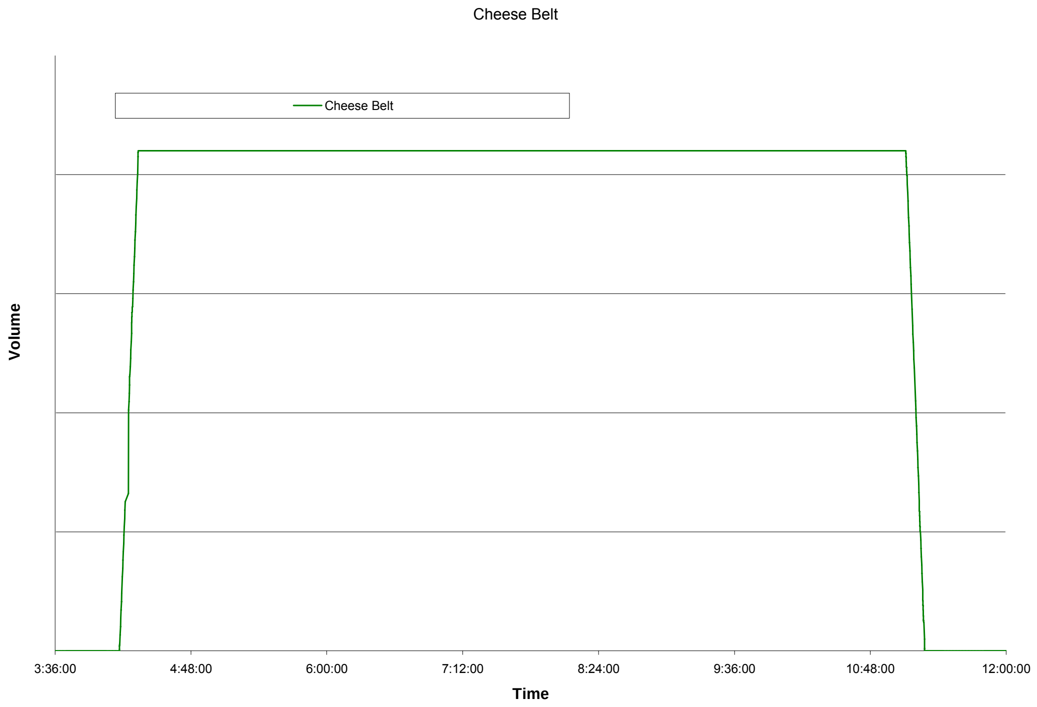


Figure 4-15 – Cheese Belt Volume Time Series

As shown in the cheese belt volume time series (Figure 4-15), the cheese belt process consists of fill and empty phases, separated by a long period of continuous flow. The cheese belt / block forming tower is modelled using a combination of capacitive units and flow units (Figure 4-16).

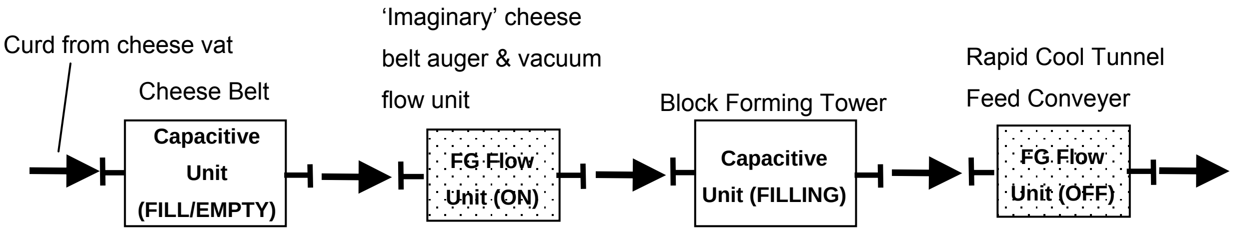


Figure 4-16 – Cheese Belt and Block Forming Tower – 'Modelled' Unit Operation Configuration

The Cheese Belt's FILLING stage models the continuous addition, at constant flow rate, of curd onto the belt from the cheese vat at the start of a processing run.

Once the Cheese Belt's capacitive unit's maximum material capacity is reached (simulating the first-added curd reaching the end of the belt), the Cheese Belt changes to the FILL/EMPTY state, the cheese belt auger and vacuum flow unit is switched ON, and curd is transferred to the block forming tower at the same flowrate as the cheese vat emptying flow rate. Thus continuous flow behaviour through a capacitive unit is modelled.

The Block Forming Tower is also a continuous flow capacitive unit, so once it reaches capacity, it will switch into the FILL/EMPTY state and the Rapid Cool Tunnel Feed Conveyer will switch into the ON state.

Figure 4-16 is the implemented configuration of the model, but not the configuration of the real process. The 'imaginary' cheese belt auger & vacuum flow unit does not in reality exist. In the real process, flow from the Cheese Belt to the Block Forming Tower is generated by a vacuum in the Block Forming Tower. See section 7.14.2 for further discussion of this situation.

4.8 Energy Transfer

The feasibility of using the Energy Port – Energy object as the mechanism for energy transfer (c.f. Material and Material Port objects) is tested in this work.

The multiple flow unit pasteurizer model shown in Figure 4-13 was constructed. This model connects two separate process models using an energy port.

The cold- and hot-side flow units' configuration and behaviour are as follows:

- Cold side flow unit inlet (port 1) and outlet (port 2) material temperatures of 3 ºC and 32 ºC respectively.
- Hot-side flow unit inlet (port 4) and outlet (port 6) material temperatures of 120 ºC and 100 ºC respectively.
- When the cold-side flow rate is non zero 0 and the flow unit detects different inlet and outlet material temperatures it seeks energy from its

inlet energy ports to (in this case) increase the outlet material port's temperature by the desired amount.

- The cold side's inlet energy port attempts to retrieve energy from its connected outlet port (the hot side).
- When the hot-side's flow unit detects an attempt to retrieve an amount of energy from its outlet energy port, it starts itself (i.e. changes its state to ON). The flow rate is calculated:

  $$\dot{m} = \frac{\Delta \mathrm{E}}{C_p \Delta T}$$

  where

  $\dot{m}$ = mass flow rate (kgs^{-1})

  C_p = specific heat ($Jkg^{-1}K^{-1}$)

  $\Delta \mathrm{E}$ = inlet and outlet ports energy difference (Js^{-1})

  ΔT = inlet and outlet ports temperature difference (K)

The amount of energy used in the pasteurization process could then be calculated from the amount of hot water used (not implemented).

This is a simplistic model of the actual pasteurization configuration used in the dairy industry. A real pasteurizer uses both intra-process energy transfer and non-process energy to heat and cool the milk (Bylund, 2003).

4.9 Unit Operation State Behaviour

Three different types of unit operation state behaviour are modelled (see section 5.1 for an example of how states are used to model unit operation behaviour).

4.9.1 Static State

A unit operation in this state remains in that state unless it is changed by user input or a result of interaction with other unit operation's in the process. The simplest example is when the unit operation it a 'ready' (or not ready) state.

For example, consider an empty cheese vat in a FILL QUEUE state. The cheese vat will remain in the fill queue unless changed externally or as a result of an expiry of a compulsory state (see section 4.9.3).

4.9.2 Dynamic State

Dynamic state behaviour causes the cheese vat to stay in a particular state for a predefined length of time, then automatically changes to another predetermined state. This mechanism allows batch modelling and automation in unit operations such as a cheese vat. It can be programmed to carry out a series of steps – each state expiring and moving the vat onto a new state.

The default state sequence for a cheese vat batch in this work is:

> FILL QUEUE **>** FILLING (until max volume reached) **>** SET (expires 30 minutes) **>** COOKING (expires 40 minutes) **>** CUTTING (expires 10 minutes) **>** EMPTY QUEUE **>** EMPTYING (until min volume reached) **>** RINSE (5 minutes) **>** FILL QUEUE

Here, the SET, COOKING, CUTTING, and RINSE states are dynamic states because they expire, then the unit operation is forced into another predefined state.

4.9.3 Compulsory State

A compulsory state must be attained within a specified time period. For example a cheese vat's CLEANING state is defined so the cheese vat is forced into that state at least once every 1440 minutes (= 24 hours).

It is always possible for the user to override the current state of a unit operation during simulation. The behaviour of different types of unit operation in different states is discussed in section 4.4.

CLEANING is also a dynamic state because once the state is attained, it will expire after a predefined period of time.

4.9.4 State Transition Mechanism

Two software tools are used to manage the transition of a unit operation out of a dynamic state or into a compulsory state. An OOP collection stores data on the times when a unit operations was last in any compulsory state, and the unit operation software object has a property which records the time the unit operation was placed in its current state. The compulsory state collection and current state property are continuously checked during simulation.

If the current simulation time (t_g) is equal to a compulsory state's pervious occurrence plus the maximum allowed time between occurrences, the unit operation will be forced into that compulsory state. If the current state is a dynamic state, and t_g is equal to the time the unit operation went into its current state plus the duration of the state, then the unit operation will be forced into the next allotted state. What this state will be is defined by the user as part of the unit operation's configuration (see section 5.1).

4.10 Simulation Timekeeping

Because a multiscale model (by definition) spans a wide time period, the implementation requires a mechanism for keeping track of, and incrementing, time. Simulation is driven using a global date/time variable (t_g) and a timer control (i.e. in this implementation Microsoft Visual Basic's timer control placed onto a PME form). At each timer 'tick', code is run which:

1. increments t_g by a predefined or pre-calculated period.
2. checks the modelling scenario for any state changes to unit operations.
3. iterates through each unit operation in the process. The code implemented depends on the state of the unit operation.
4. generates process and unit operation data.

The global date/time variable t_g keeps track of the simulation's 'actual' time and increases with each iteration by the value of the time increment.

4.10.1 A Failed Simulation Control Mechanism

The initial software implementation of Process Unit form classes included a timer control which recalculated the instantiated form’s Process Unit and motivated the simulation. However, this approach proved both unreliable and inefficient.

Firstly it was difficult to maintain an orderly iteration sequence. Each unit operation’s timer ticked independently which resulted in a mass balance error because the capacitive unit might not recalculate with the correct process data.

For example, consider a pump which when turned on changes the flow rate of an attached capacitive unit’s material port’s flowrate *after* the capacitive unit has recalculated. This occurs because the capacitive unit’s timer has (arbitrarily) ticked first (Figure 4-17). The process time increment is 1 second and the capacitive unit starts the iteration empty.

When the capacitive unit recalculates it does so without the correct flowrate at port 2 because the flow unit hasn’t yet recalculated and set the connected port flowrates. So the mass in the capacitive unit remains at zero and a mass balance error of (in this case) 50kg/s x 1s = 50kg occurs. Port 2’s flowrate should be 50 kg/s when the capacitive unit is recalculated, giving a content mass after recalculation of 50kg.

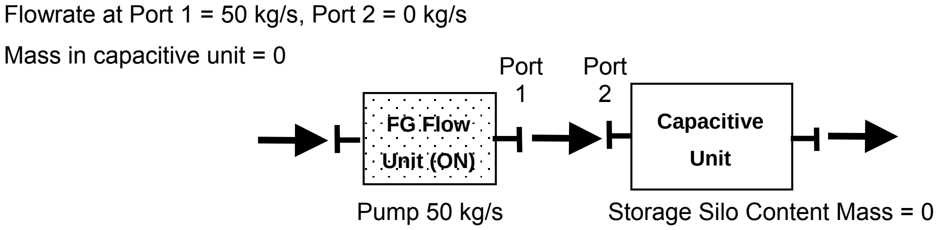

Flowrate at Port 1 = 50 kg/s, Port 2 = 0 kg/s
Mass in capacitive unit = 0
Port 1
Port 2
FG Flow Unit (ON)
Capacitive Unit
Pump 50 kg/s
Storage Silo Content Mass = 0

Figure 4-17 – Process Simulation Process Unit Recalculation Sequential Order Error

It became apparent that it was important to control the order of unit operation recalculation. Because some flow units are the driving force for material transfer, here, flow units are recalculated first, This sets all the unit operation material port’s flowrates. Only then are capacitive units recalculated.

The second problem was, as the number of unit operations in the process being modelled increased the amount of computer processing power required to simultaneously recalculate unit operations became significant. Usually the simulation slowed considerably, but sometimes it froze completely. This was especially important when small fixed increment time intervals were used (see section 4.10.2).

4.10.2 Simulation Time Increment

The user selects one of two methods of incrementing the simulation time.

The first method increments the process by a fixed time interval with each timer tick. If the user sets the increment at (say) 10 seconds, each increment of the controlling timer increments the process by 10 seconds. i.e.

Timer Tick	Simulation 'Actual' Time (t_g)
0	1 January 2005 12:00:00am (Start Time)
1	1 January 2005 12:00:10am
2	1 January 2005 12:00:20am
3	1 January 2005 12:00:30am
4	1 January 2005 12:00:40am

The second method examines the process to find what the next state of each unit operation will be, and the time increment that would implement that state. With a one exception, the smallest time increment returned from all unit operations is used as the time increment for the next iteration. The exception is when the expiry time of the production scenario (i.e. the time to midnight) is less than the smallest time increment. In that case the time increment is the seconds to midnight.

Each iteration can (and generally will) have a different time increment. i.e.

Timer Tick	Increment (s)	Simulation 'Actual' Time (t_g)
0		1 January 2005 12:00:00am (Start Time)
1	120	1 January 2005 12:02:00am
2	60	1 January 2005 12:03:00am
3	240	1 January 2005 12:07:00am
4	1980	1 January 2005 12:40:00am

This fixed-interval time increment method produces redundant data, as well as a slower simulation solution and than the discontinuity method (see section 7.6 for a full discussion of the discontinuity method). The smallest time increment currently possible is one second (this is a software implementation limitation).

4.10.3 Daily Production Model Iteration

The simulation's actual time t_g is used to select and increment the production scenario, which controls unit operation behaviour during the simulation's 'current' 1 day cycle. Each unit operation is iterated and recalculated, the process time is incremented, and data is generated. Because t_g is a unique incrementing variable, it can be used as part of a unique identification key for process simulation data. Data can also be easily compared to historic plant data or current process data using t_g.

4.10.4 Simulation Speed

Simulation speed is controlled by changing the timer 'tick' interval. Increasing the interval slows the timer, while decreasing the interval causes it to tick more frequently. This means the user can change the model iteration rate. For example, a 1ms timer tick interval = 1000 iterations per second; a 50ms tick interval = 20 iterations per second; 1s tick interval = 1 iteration per second.

5 Model Operation

5 steps are required to design and configure the process for simulation:

1. Define unit operation type templates, their properties and state collection.
2. Create individual unit operations based on unit operation type templates.
3. Connect the unit operations into a process flowsheet.
4. Construct 1 day production scenarios, and add these in sequential order to construct a modelling scenario.
5. Set the simulation's start date/time, the time increment of each timer step, and the timer 'tick' interval.

The CD included with this thesis has a copy of the software application used to perform simulations. Appendix A provides the necessary installation and operating instructions to view a preconfigured simulation.

In this section, the Process Modelling Environment (PME) implementation is shown. Here, the PME manifests as several software forms; the Process Modelling Executive Form, the Unit Operation Type Template Form, and the Modelling Scenario Form, all of which are discussed below.

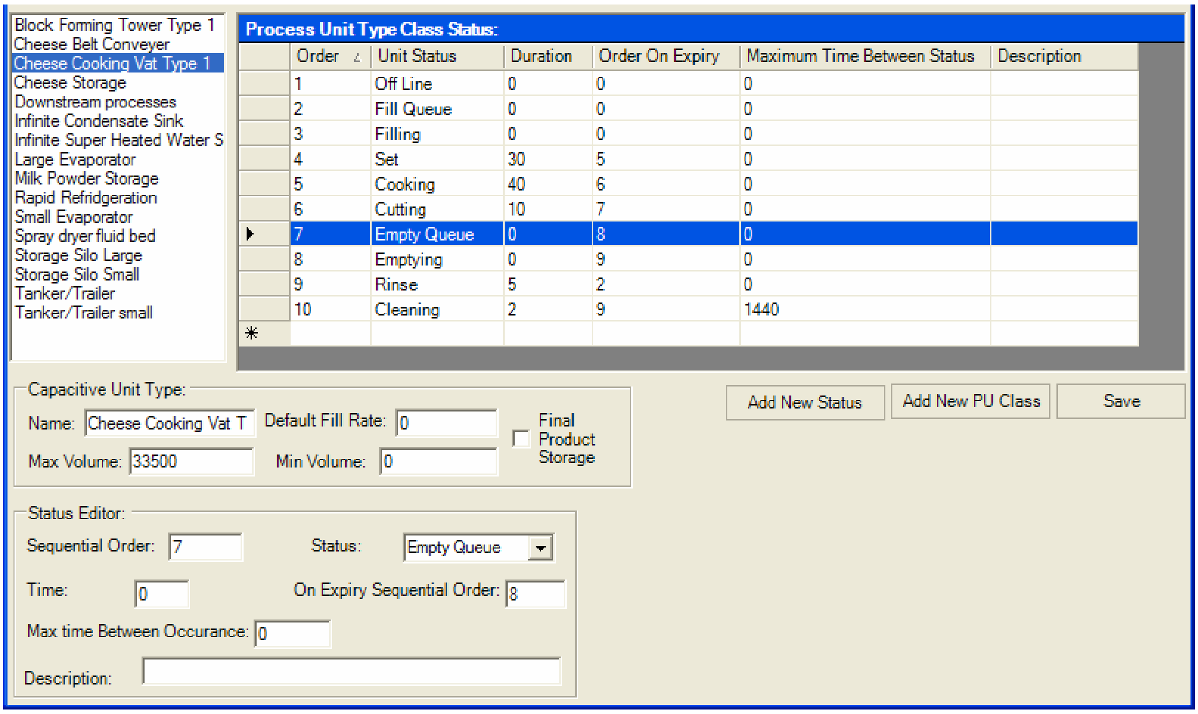
5.1 Unit Operation Type Templates

The unit operation's *type template* determines its behaviour and is the basis for the creation of individual unit operation partial models. They are essentially the model for the unit operation. The template consists of properties (e.g. mass or volume capacity) and a state collection. Figure 5-1 shows the template for a capacitive unit. The capacitive type name is Cheese Vat Type 1, its volume is 33,500L). A cheese vat modelled on this template has 10 possible states (i.e. its state collection count = 10).

As identified in section 4.3.2 a unit operation will exist in different states at different times. By controlling the state of the unit operation its behaviour is determined. A unit operation's state collection provides the list of possible states.

Instructions for some of the cheese vat's behaviour is obtained from the state collection. Its behaviour is determined by changing its state using one of two control mechanisms:

1. The state changes automatically as it steps through a predefined sequence of states.
2. The user can set the state of a cheese vat during simulation, using the production scenario state control mechanism (discussed in section 4.3.5).



Process Unit Type Class Status:

	Order	Unit Status	Duration	Order On Expiry	Maximum Time Between Status	Description
	1	Off Line	0	0	0	
	2	Fill Queue	0	0	0	
	3	Filling	0	0	0	
	4	Set	30	5	0	
	5	Cooking	40	6	0	
	6	Cutting	10	7	0	
▶	7	Empty Queue	0	8	0	
	8	Emptying	0	9	0	
	9	Rinse	5	2	0	
	10	Cleaning	2	9	1440	
*						

Capacitive Unit Type:

Name: Cheese Cooking Vat T Default Fill Rate: 0 □ Final Product Storage

Max Volume: 33500 Min Volume: 0

Status Editor:

Sequential Order: 7 Status: Empty Queue

Time: 0 On Expiry Sequential Order: 8

Max time Between Occurance: 0

Description:

Figure 5-1 – Cheese Vat Template

Figure 5-1 shows the unit operation template for a cheese cooking vat. The state collection gives a cheese vat created from this template the following behaviour:

1. The first state (order = 1) is OFF LINE. Therefore, when the cheese vat object is first instantiated by the process modelling environment, its state is set to OFF LINE.

2. The FILL QUEUE state does not expire (duration = 0).
3. Upon completion of the cheese vat FILLING (order = 3), it will move into the SET state (order = 4).
4. The cheese vat will remain in the SET state for 30 minutes, then move into the COOKING state (duration = 30, order on expiry = 5).
5. The cheese vat will remain in the COOKING state for 40 minutes, then move into the CUTTING state (duration = 40, order on expiry = 6).
6. The cheese vat will remain in the CUTTING state for 10 minutes, then move into the EMPTY QUEUE state (duration = 10, order on expiry = 7).
7. Upon completion of the cheese vat EMPTYING (order = 8), it will move into the RINSE state (order = 9).
8. The cheese vat will remain in the RINSE state for 5 minutes, then move into the FILL QUEUE state (duration = 5, order on expiry = 2).
9. If the cheese vat is not cleaned for 1440 minutes, it will go into the CLEANING state for 2 minutes then move into the RINSE state (duration = 2, order on expiry = 9).

The mechanism initiating these state changes was discussed in section 4.9.4.

5.1.1 Unit Operation Types Created

15 unit operation type models were developed in this work to model the cheese making process:

Capacitive: Raw milk storage silo, cream storage silo, cheese milk silo, cheese vat, whey collector, block forming tower, rapid cool tunnel, cheese storage.

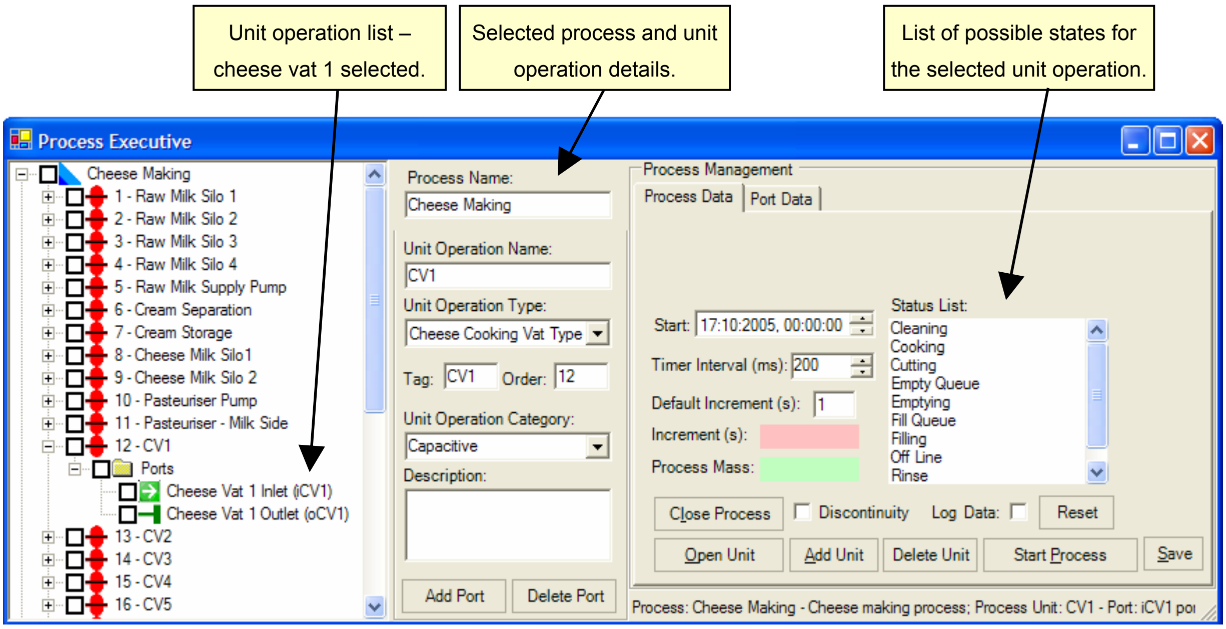
Flow: Raw milk supply pump, cream separator, pasteurizer, pasteurizer pump, cheese belt, rapid cool tunnel conveyer, cheese transfer to storage.

In addition, a spray dryer process was configured to enable testing of the Process Modelling Environment. Because this was outside the parameters of this work it will not be discussed further.

5.2 Creating a Unit Operation

In this implementation, unit operations are created within the context of the process. The following steps are performed:

1. A Process object is created.
2. Each unit operation is created and added to the Process.
3. The unit operation is defined (based on a unit operation template – see section 5.1).
4. Ports are added to the unit operation. Each port's type is defined (i.e. material, energy, or information port), given a direction (i.e. inlet, outlet, inlet/outlet) and configured depending on the its type. For example, a material type port will be given a material.



Unit operation list – cheese vat 1 selected.

Selected process and unit operation details.

List of possible states for the selected unit operation.

Process Executive

- Cheese Making
  - 1 - Raw Milk Silo 1
  - 2 - Raw Milk Silo 2
  - 3 - Raw Milk Silo 3
  - 4 - Raw Milk Silo 4
  - 5 - Raw Milk Supply Pump
  - 6 - Cream Separation
  - 7 - Cream Storage
  - 8 - Cheese Milk Silo1
  - 9 - Cheese Milk Silo 2
  - 10 - Pasteuriser Pump
  - 11 - Pasteuriser - Milk Side
  - 12 - CV1
    - Ports
      - Cheese Vat 1 Inlet (iCV1)
      - Cheese Vat 1 Outlet (oCV1)
  - 13 - CV2
  - 14 - CV3
  - 15 - CV4
  - 16 - CV5

Process Name: Cheese Making

Unit Operation Name: CV1

Unit Operation Type: Cheese Cooking Vat Type

Tag: CV1 Order: 12

Unit Operation Category: Capacitive

Description:

Add Port | Delete Port

Process Management

Process Data | Port Data

Start: 17:10:2005, 00:00:00

Timer Interval (ms): 200

Default Increment (s): 1

Increment (s):

Process Mass:

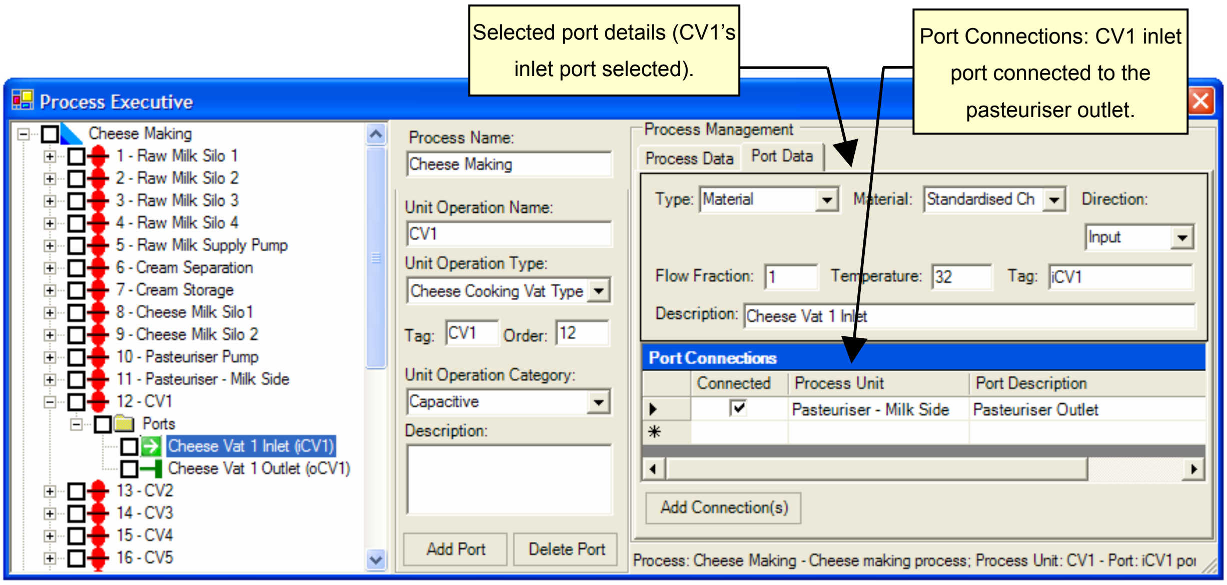
Status List:
Cleaning
Cooking
Cutting
Empty Queue
Emptying
Fill Queue
Filling
Off Line
Rinse

Close Process | □ Discontinuity | Log Data: □ | Reset

Open Unit | Add Unit | Delete Unit | Start Process | Save

Process: Cheese Making - Cheese making process; Process Unit: CV1 - Port: iCV1 poi

Figure 5–2a – Process Modelling Executive - Process Data Tab



Selected port details (CV1's inlet port selected).

Port Connections: CV1 inlet port connected to the pasteuriser outlet.

Process Executive

- Cheese Making
  - 1 - Raw Milk Silo 1
  - 2 - Raw Milk Silo 2
  - 3 - Raw Milk Silo 3
  - 4 - Raw Milk Silo 4
  - 5 - Raw Milk Supply Pump
  - 6 - Cream Separation
  - 7 - Cream Storage
  - 8 - Cheese Milk Silo1
  - 9 - Cheese Milk Silo 2
  - 10 - Pasteuriser Pump
  - 11 - Pasteuriser - Milk Side
  - 12 - CV1
    - Ports
      - Cheese Vat 1 Inlet (iCV1)
      - Cheese Vat 1 Outlet (oCV1)
  - 13 - CV2
  - 14 - CV3
  - 15 - CV4
  - 16 - CV5

Process Name: Cheese Making

Unit Operation Name: CV1

Unit Operation Type: Cheese Cooking Vat Type

Tag: CV1 Order: 12

Unit Operation Category: Capacitive

Description:

Process Management

Type: Material Material: Standardised Ch Direction: Input

Flow Fraction: 1 Temperature: 32 Tag: iCV1

Description: Cheese Vat 1 Inlet

Port Connections

Connected	Process Unit	Port Description
☑	Pasteuriser - Milk Side	Pasteuriser Outlet

Process: Cheese Making - Cheese making process; Process Unit: CV1 - Port: iCV1 poi

Figure 5-2b - Process Modelling Executive – Port Tab

Figure 5-2a shows the process modelling executive form – process data tab. Figure 5-2b shows the same form with the port tab displayed. The unit operations which constitute the cheese making process are in the left hand column, with the detail for the selected cheese vat (name = CV1) displayed in the central column.

5.3 Connecting Unit Operations into the Process Flowsheet

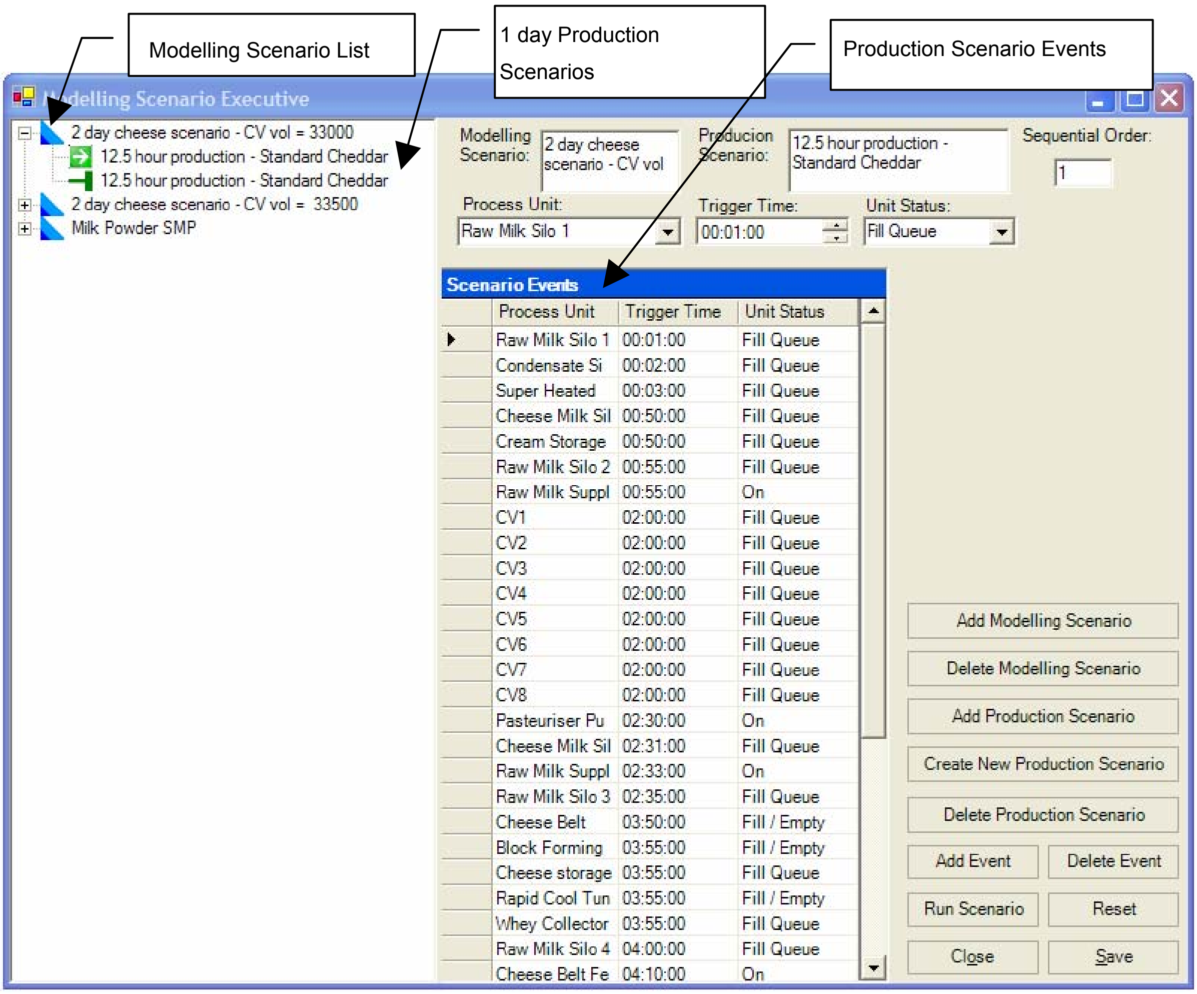
Once the process' unit operations are defined, they are connected together into a process flowsheet. This requires the interconnection of ports. The following rules are defined:

1. A unit operation cannot connect to itself (i.e. a port on a unit operation cannot connect to another port on the same unit operation).
2. A port cannot connect to a port of the same direction (e.g. inlet ports can only connect to outlet or inlet/outlet ports).
3. Ports can only connect to ports of the same type (i.e. a material port can only connect to another material port).
4. A material port can only connect to a port which has the same material.

The PortConnection object is used as the mechanism for connecting ports. A PortConnection consists of an inlet port, and an outlet port.

5.4 Creating a Modelling Scenario

After a process' unit operations are created and the process constructed, the user creates a modelling scenario.


Modelling Scenario List
1 day Production Scenarios
Production Scenario Events
Modelling Scenario Executive
Add Modelling Scenario
Delete Modelling Scenario
Add Production Scenario
Create New Production Scenario
Delete Production Scenario
Add Event
Delete Event
Run Scenario
Reset
Close
Save

- 2 day cheese scenario - CV vol = 33000
  - 12.5 hour production - Standard Cheddar
  - 12.5 hour production - Standard Cheddar
- 2 day cheese scenario - CV vol = 33500
- Milk Powder SMP

Modelling Scenario: 2 day cheese scenario - CV vol

Producion Scenario: 12.5 hour production - Standard Cheddar

Sequential Order: 1

Process Unit: Raw Milk Silo 1

Trigger Time: 00:01:00

Unit Status: Fill Queue

Scenario Events

Process Unit	Trigger Time	Unit Status
Raw Milk Silo 1	00:01:00	Fill Queue
Condensate Si	00:02:00	Fill Queue
Super Heated	00:03:00	Fill Queue
Cheese Milk Sil	00:50:00	Fill Queue
Cream Storage	00:50:00	Fill Queue
Raw Milk Silo 2	00:55:00	Fill Queue
Raw Milk Suppl	00:55:00	On
CV1	02:00:00	Fill Queue
CV2	02:00:00	Fill Queue
CV3	02:00:00	Fill Queue
CV4	02:00:00	Fill Queue
CV5	02:00:00	Fill Queue
CV6	02:00:00	Fill Queue
CV7	02:00:00	Fill Queue
CV8	02:00:00	Fill Queue
Pasteuriser Pu	02:30:00	On
Cheese Milk Sil	02:31:00	Fill Queue
Raw Milk Suppl	02:33:00	On
Raw Milk Silo 3	02:35:00	Fill Queue
Cheese Belt	03:50:00	Fill / Empty
Block Forming	03:55:00	Fill / Empty
Cheese storage	03:55:00	Fill Queue
Rapid Cool Tun	03:55:00	Fill / Empty
Whey Collector	03:55:00	Fill Queue
Raw Milk Silo 4	04:00:00	Fill Queue
Cheese Belt Fe	04:10:00	On

Figure 5-3 – 2 Day Modelling Scenario: Standard Cheddar

Figure 5-3 shows a 2 day cheese modelling scenario, which consists of two 1 day production scenarios. In this case the production scenarios are the same – i.e. the process repeats the same production scenario for each day of the modelling scenario. The 1 day production scenario is essentially a collection of controlling instructions to individual unit operations, which set the state of the unit operation at a particular time within the 1 day period.

In Figure 5-3, the *12.5 hour production – Standard Cheddar* production scenario is selected. This production scenario, for example, places the raw milk silo unit operation into a FILL QUEUE at 00:01:00. The raw milk supply pump is switched ON at 00:55:00, while the Cheese Vats (CV1 – CV8) are placed into the FILL QUEUE at 02:00:00 (note: in this example the modelling scenario begins at 00:00:00 on 17/10/2005 – see Figure 5-2a).

The model is tested during construction by running the simulation using a 1 second fixed-interval time increment. This ensures at least 1 feasible model solution is guaranteed.

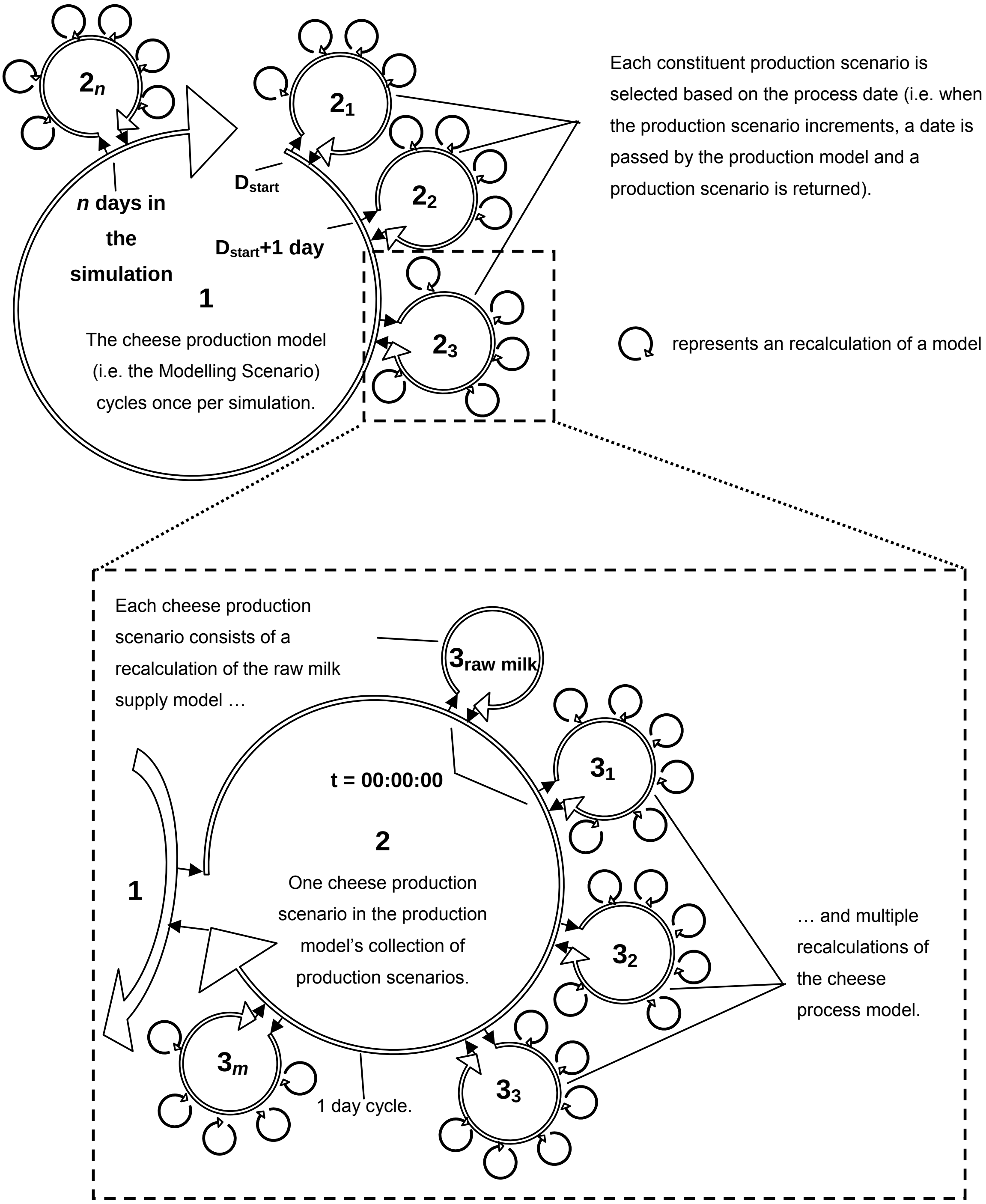
The simulation is completed when the modelling scenario's final production scenario has been run. During simulation, individual unit operation and production data is being generated.

5.5 Simulation Solution

The multiscale model's simulation is solved by time stepping. The simulation is given a process start date ($\mathbf{D_{start}}$). With each recalculation, the process time is incremented by some specified time period. The size of the time step is either set manually or is generated from the process scale or the unit operation scale partial model (previously discussed in section 4.10).

Figure 5-4 represents a production model simulation (a *modelling scenario)*. A single cycle of the modelling scenario represents a complete production simulation (Figure 5-4 - **1**). It consists of a collection of production scenarios. Each of the production scenarios which constitute the modelling scenario are used to control the cheese making process (Figure 5-4 - $\mathbf{2}_{\mathbf{1\ to}\ \boldsymbol{n}}$ where *n* = number of production scenarios = the number of days of the simulation).

Individual production scenarios represent a 1 day cycle (Figure 5-4 - **2)** of the production model's simulation. For each production scenario, the raw milk supply model is recalculated once (i.e. a date is passed to the raw milk supply model and an amount of milk which is available to the model for that day is returned).

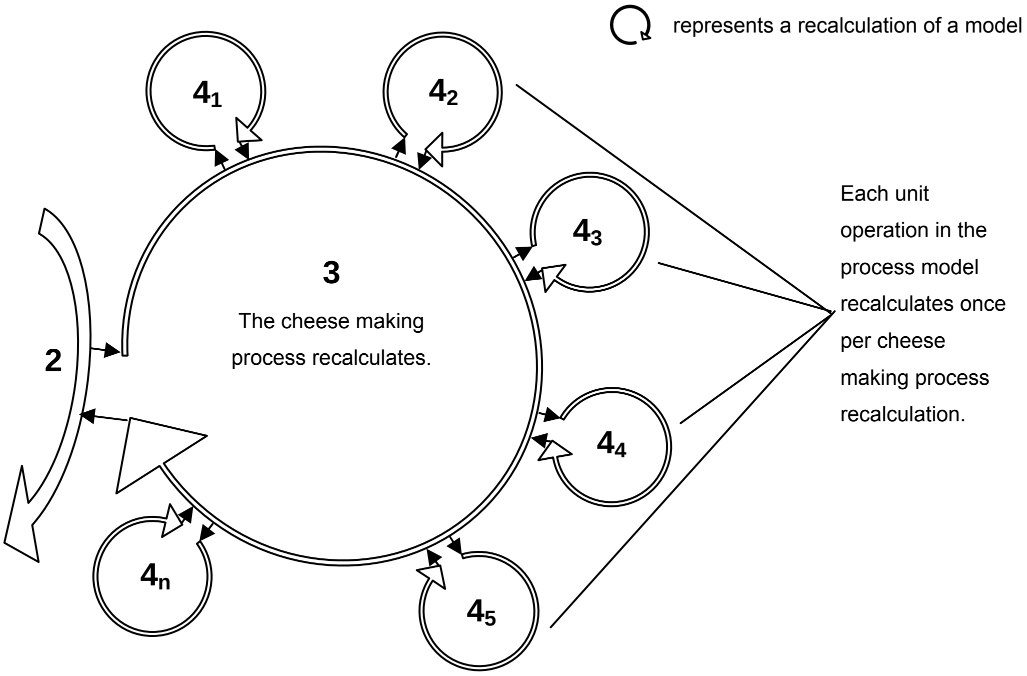


2_n
2_1
2_2
2_3
Each constituent production scenario is selected based on the process date (i.e. when the production scenario increments, a date is passed by the production model and a production scenario is returned).
n days in the simulation
D_{start}
D_{start}+1 day
1
The cheese production model (i.e. the Modelling Scenario) cycles once per simulation.
represents an recalculation of a model
Each cheese production scenario consists of a recalculation of the raw milk supply model …
$3_{raw\ milk}$
t = 00:00:00
3_1
2
One cheese production scenario in the production model's collection of production scenarios.
1
3_2
… and multiple recalculations of the cheese process model.
3_m
1 day cycle.
3_3

Figure 5-4 – Cheese Production Model Iteration

The process model recalculates multiple times (Figure 5-4 – *m* iterations, each iteration $\mathbf{3}_{1\ \mathrm{to}\ m}$ where *m* = the number of times the process model iterates).

The raw milk supply model is recalculated at time t = 00:00:00 (Figure 5-4 – $\mathbf{3}_{\mathbf{raw\ milk}}$) of the day being modelled. The first recalculation of the cheese process model also occurs at this time (for clarity it is shown offset in Figure 5-4 - $\mathbf{3}_{\mathbf{1}}$). Subsequent recalculations of the cheese making process occur with each time increment (section 4.10).



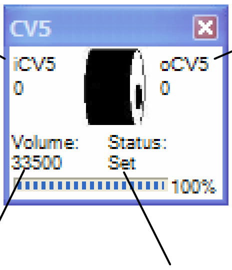
represents a recalculation of a model
4_1
4_2
4_3
3
The cheese making process recalculates.
2
4_4
4_n
4_5
Each unit operation in the process model recalculates once per cheese making process recalculation.

Figure 5-5 – Cheese Process Model Recalculation

For each recalculation of the process model (Figure 5-5 - **3**) all unit operations are recalculated once (Figure 5-5 – $\mathbf{4}_{\mathbf{1\ to\ n}}$ where n = the number of unit operations in the process). A capacitive unit operation's material content microscale partial model will recalculate if certain criteria are met (discussed in section 7.6).

5.6 Monitoring the Simulation

The implementation of process graphics is rudimentary, using forms. The graphical user interface for monitoring a unit operation during simulation is the unit operation form. The user can instantiate individual unit operations, and relocate them around their container MDI form. Two generic form objects are used. The Capacitive Unit form (Figure 5–6a), and the Flow Unit form (Figure 5–6b).



CV5

iCV5
0

oCV5
0

Volume: 33500

Status: Set

100%

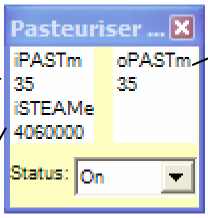
Inlet material port:
Tag = iCV5
Port flowrate = 0kg/s

Outlet material port:
Tag = oCV5
Port flowrate = 0kg/s

Current volume = 33500 L (100%)

State = SET (note: this graphic shows a cheese vat)

Figure 5-6a - Capacitive Unit Form



Pasteuriser ...

iPASTm
35

oPASTm
35

iSTEAMe
4060000

Status: On

Inlet material port:
Tag = iPASTm
Port flowrate = 35kg/s

Outlet material port:
Tag = oPASTm
Port flowrate = 35kg/s

Inlet energy port:
Tag = iSTEAMe
Port energy flow = 4060000 Joules

State = ON (note: this graphic shows a pasteuriser)

Figure 5-6b – Flow Unit Form

Both form objects display unit operation name, state, and port information. The capacitive unit form also displays volume information. Using these forms, the user can construct a visual representation of the process, and monitor each unit operation’s behaviour during simulation. No visual representation of the connections and flow between unit operations was implemented.

Figure 5-7 shows the user view of the cheese making process during simulation.

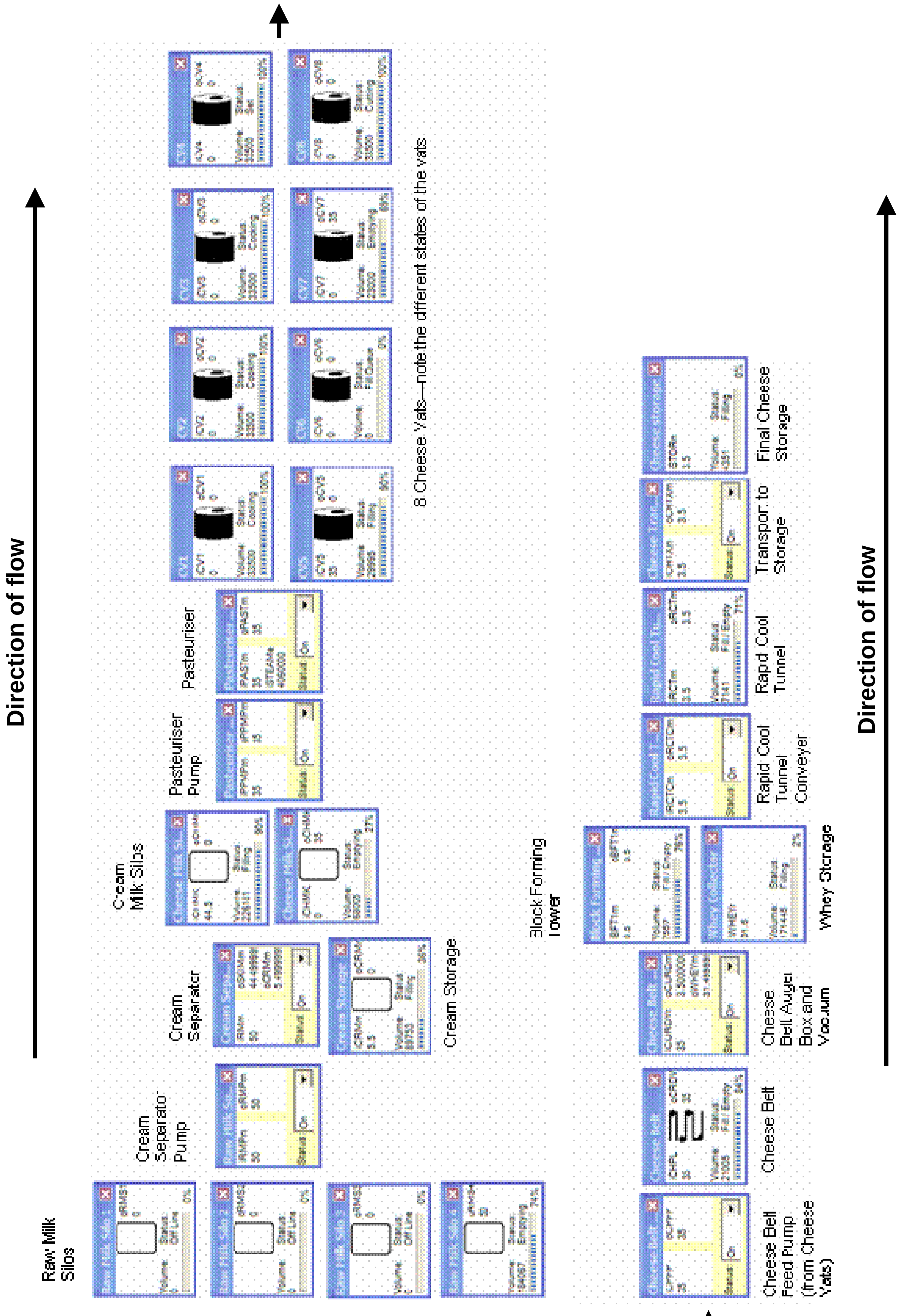


Figure 5-7 – Cheese Making Simulation (User View)

6 Model Results, Accuracy and Verification

The multiscale model should be tested against a real cheese making process to be properly tested and verified. In the absence of that here, the model was verified using:

- a simple overall mass balance
- unit operation volume time series graphs
- unit operation state Gantt charts
- manufactured product time series graphs
- raw milk consumption time series graphs

To test the multiscale model, several simulations were conducted using a combination of:

- different modelling scenarios (i.e. different combinations of 1 day production scenarios),
- different time periods (i.e. 1 day, 2 day, 6 day),
- the fixed-interval time increment method using different iteration time intervals,
- the discontinuity time increment method.

As discussed in section 5.4 a production scenario is tested during construction with a 1 second fixed-interval time increment. This ensures that the modelling problem has at least one feasible solution (albeit an inefficient one).

For all production scenarios tested, once the 1 second time increment solution was found, the model reached a successful solution when running in discontinuity time increment mode.

6.1 Data Generated

The cheese production model presented here generated data on capacitive unit material content and state, flow unit state, totalized manufactured product, totalized energy consumption, and raw material use data. Each new data record receives a date/time stamp.

A simulation running in fixed-increment mode of 1 second time increments completes 43,200 steps in 12 hours. The 20 capacitive unit operations in the cheese making process modelled here produced 864,000 data records for a 12 hour simulation – for volume alone.

The same simulation operating in discontinuity mode produced 200 records per capacitive unit for 12 hours simulation. For 20 capacitive unit operations, this equates to 4000 data records for volume.

6.2 Overall Mass Balance

The mass balance is simply the sum of the material contents within each capacitive unit operation, calculated with each iteration of the multiscale model. 1,000,000kg per day of raw milk was made available as feed for the simulation.

As the fixed-interval time increment was increased, the solution time decreased, and the overall mass balance error trended upwards. Table 6-1 shows the results for 1 day cheese making process simulations processing 1,000,000 kg of raw milk. All simulations ran with a timer speed of 10 steps per second (the same simulation in discontinuity mode is shown for comparison).

Table 6-1 – Fixed Interval Simulation

Fixed Interval Time Increment	**Simulation Duration (seconds)**	**Overall Mass Balance Error**
1s	3600	0%
2s	1800	+0.012%
3s	1200	+0.039%
4s	900	+0.027%
5s	720	+0.048%
6s	600	+0.133%
10s	360	+0.048%
12s	300	+0.042%
15s	270	+0.234%
20s	180	+0.153%
30s	120	+0.506%
60s	60	+1.011%
Discontinuity	***90***	***0%***

6.3 Time Series Graphs

While the model mass balance gives one indication of the validity of the result, it doesn’t provide unit operation detail.

Figure 6-1 shows the volume time series for cheese vat 8 from a cheese making simulation (in fixed-interval time increment mode). This graph has all the features expected from a cheese vat:

- The volume went through repeated fill and empty cycles,
- The linearity of the fill and empty stages indicates constant fill and empty flowrates,
- The mirror image fill and empty line slopes indicate the fill flowrate was the same as the empty flowrate,

- The maximum volume stage for each cycle is the same duration, indicating a batch cycle.

Time series graphs for other unit operations are shown in Appendix B. There is little difference in data quality between the fixed and discontinuity time increment modes simulation results. All the key behaviors of process units were reproduced.

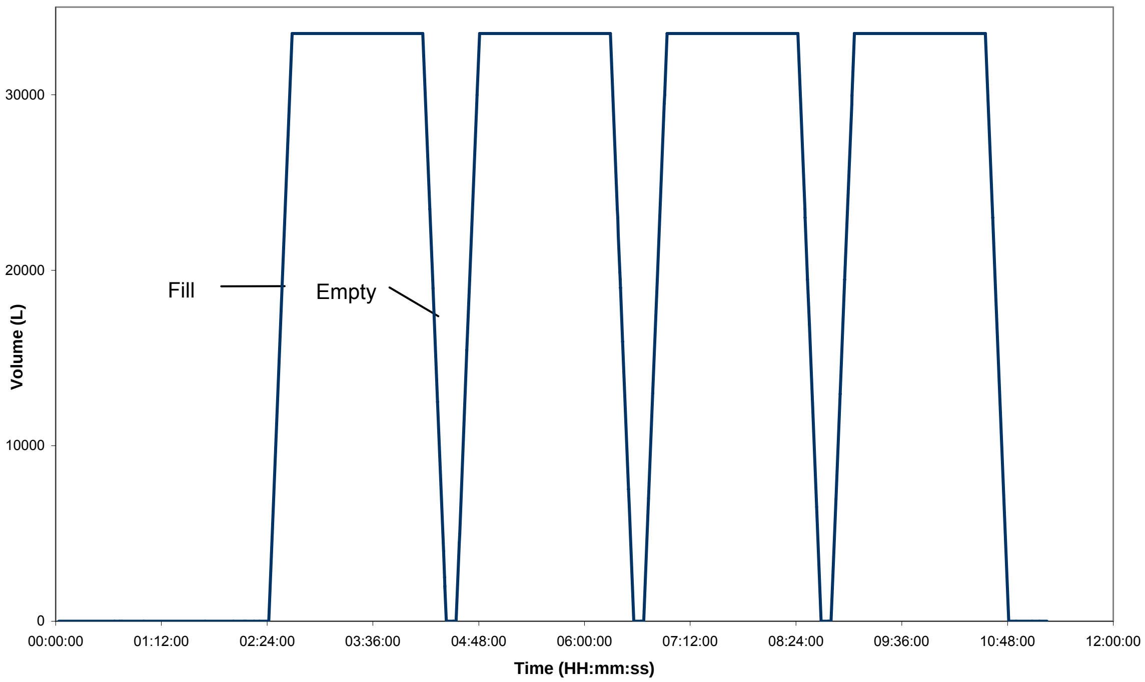


Figure 6-1 – Volume Time Series for Cheese Vat 1

6.4 Gantt Charts

Gantt charts are useful for providing unit operation utilization information. Figure 6-2 shows a unit operation state Gantt chart for the 8 cheese vats in an 12 hour cheese making simulation. It shows the following:

- the batch cycle nature of each cheese vat. Cheese vat 1 fills, then enters the SET state, followed by the COOKING state, then CUTTING. At the completion of CUTTING, it enters the EMPTY QUEUE, then begins

EMPTYING. Once emptied, cheese vat 1 is washed using a RINSE, then re-enters the FILL QUEUE.

- the sequential batch nature of the combination of the cheese vats is shown. Cheese vat 1 FILLING occurs, followed by cheese vat 2, then cheese vat 3 and so on until cheese vat 8. Each of the vats then enters its sequence of states. The time lag for each cheese vat's sequence is equivalent to the time required to fill a cheese vat.

The Gantt chart demonstrates the ability of the multiscale model presented here to accurately model the changing utilization of individual unit operations.

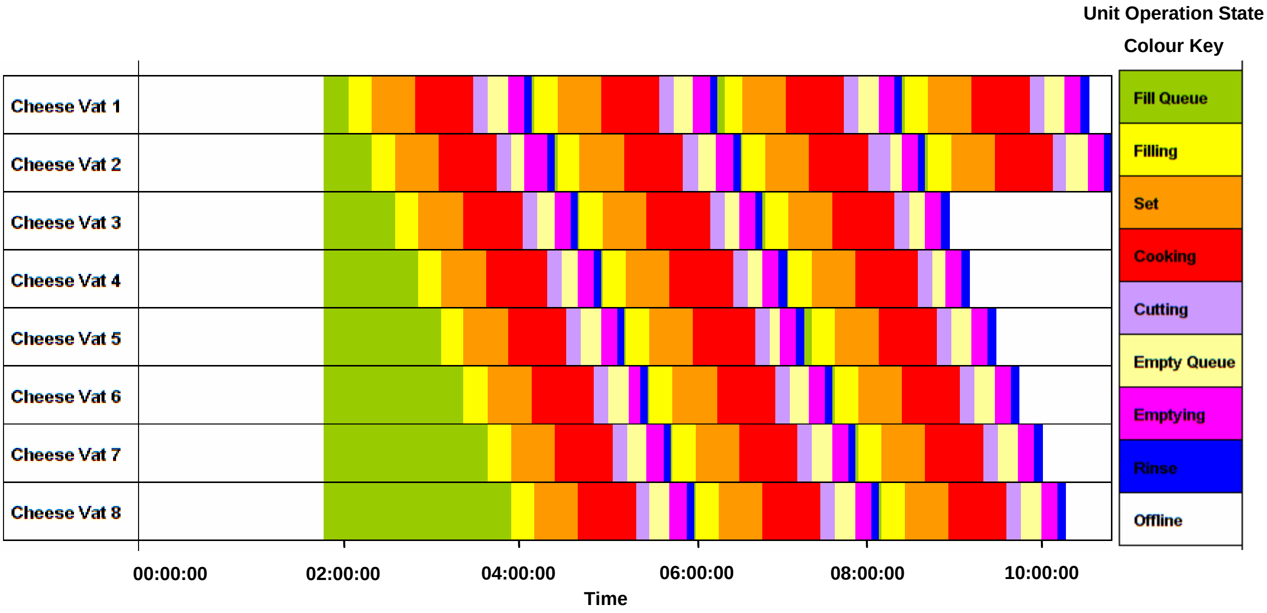


Figure 6-2 – 8 Cheese Vats Gantt Chart – 12 Hour Simulation

Figure 6-2 was generated using a Microsoft Excel spreadsheet.

6.5 Manufactured Product

A manufactured products collection class stores all the manufactured product in final shipping form. The production model was set up to manufacture a product called 25kg Bulk Cheddar.

The manufactured product collection gains a 25kg Bulk Cheddar object each time 25kg of product is added to the manufactured product storage unit operation. The manufactured product collection is a time based dataset which can be used by adjacent macroscale partial models (see section 7.5.2).

Figure 6-3 shows the time series data for the production of 25kg Bulk Cheddar. Day 1 produced 3484 x 25kg units. Day 2 produced 3618 x 25kg units, taking the total number of 25 kg units to 7102. These unit amounts came from the 87103kg of cheese produced on day 1 and 90453kg produced on day 2.

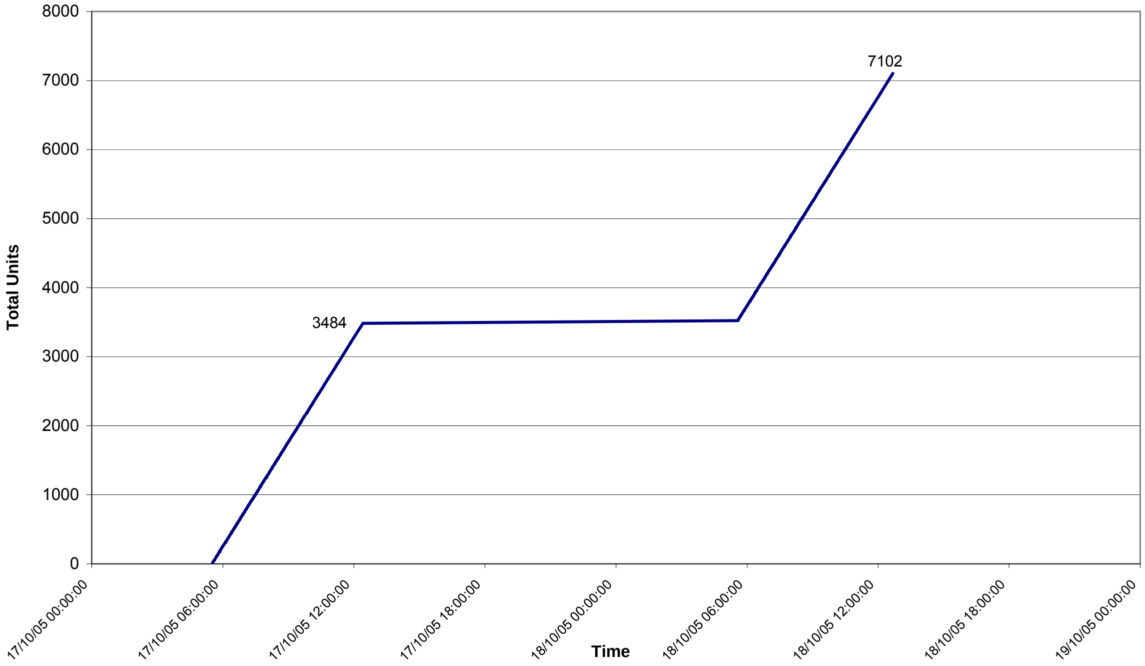


Figure 6-3 – Total Manufactured Units of 25kg Bulk Cheddar – 2 day simulation

The different amounts produced on each day reflects the extra cheese cooking vat batch which was done on day 2 as a result of the day 1 run leaving some unprocessed cheese milk. The remaining milk was not enough for an extra batch on day 1, but combined with the day 2 cheese milk was enough for the extra batch.

Figure C-1 in appendix C shows the manufactured unit graph for a 6 day simulation.

6.6 Raw Material Consumption

Figure 6-4 shows a graph of the consumption of raw milk as it is used by the cheese making process. The simulation makes an amount of raw milk available to all processes being modelled, though here only the cheese making process consumes the milk. The amount that is made available is obtained on each 1 day iteration from the milk curve.

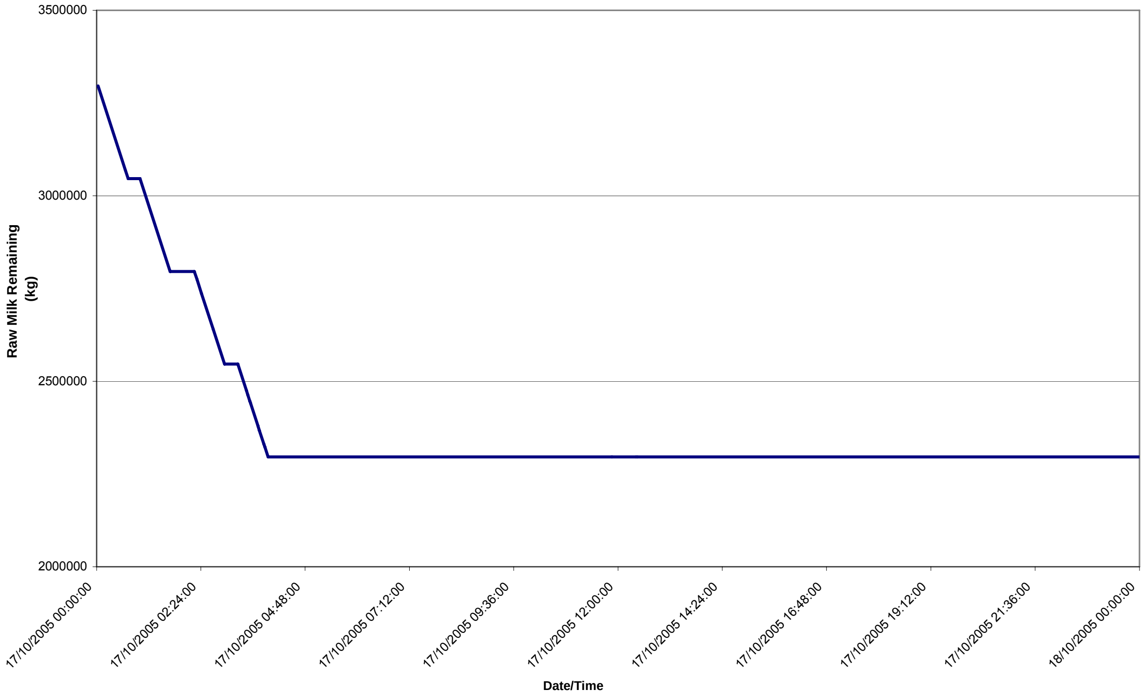


Figure 6-4 –Raw Milk Consumption –1 Day Simulation

On Figure 6-4, 3,296,422 kg is available at the start of the simulation (at 00:00:00 on the 17/10/2005). As the cheese making process consumes the raw milk, the total available amount reduces. The final amount of milk is taken at about 04:00:00, after which the amount available remains constant. This amount is 2,296,422 kg, which is 1,000,000 kg less that the amount available at the start of the day (and exactly the amount consumed by the cheese making process).

Figure 6-5 shows the raw milk consumption for a 6 day simulation. On each 1 day iteration a new (and in this case slightly increased) amount of raw milk is made available to the simulation from the milk curve.

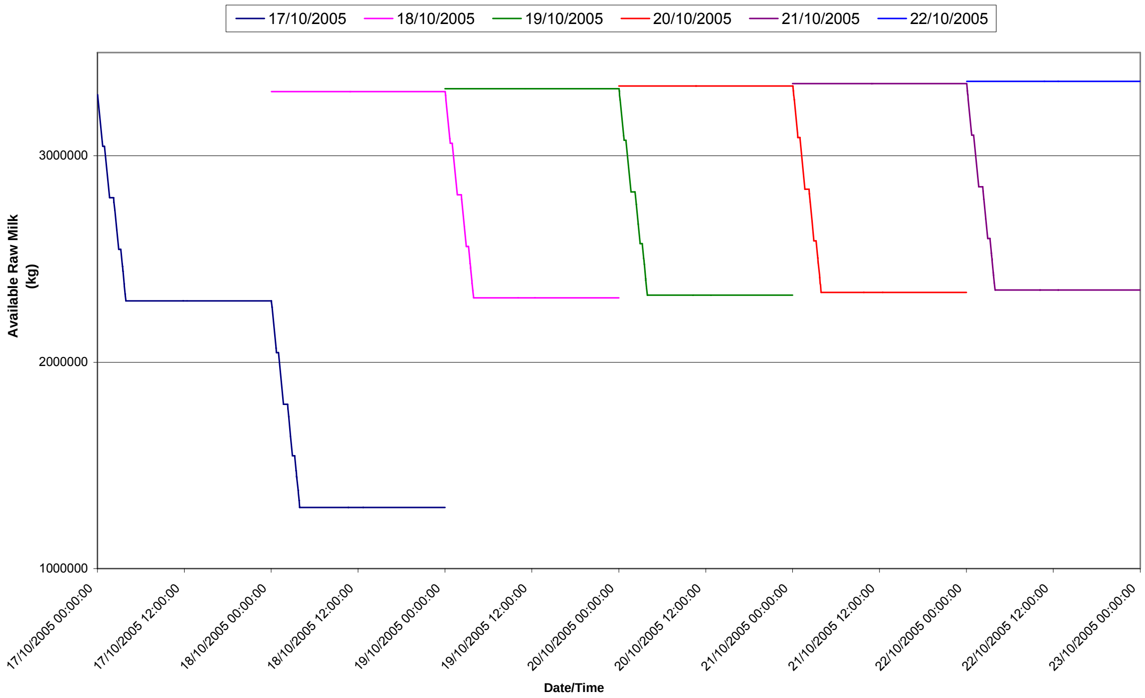


Figure 6-5 –Raw Milk Consumption – 6 Day Simulation

Two important dairy manufacturing business rules which are modelled are evident from Figure 6-5:

1. Raw milk 'expires' after an amount of time has passed from it being added to the model. In this simulation the chosen expiry time is 48 hours, after which any unused milk is no longer available for processing. This models the perishable nature of raw milk.

2. The oldest milk is processed first. The raw milk from 17/10/05 was used on that date. On the 18/10/05 a new supply of raw milk was made available, but was not used because older milk was still available (i.e. the remaining milk from 17/10/05). On the 19/10/05 the raw milk from the 18/10/05 was used, and so on.

Figure 6-5 implies a large amount of raw milk having to be disposed of because it is unprocessed by its expiry date. In reality most or all of the raw milk would be processed using other processing facilities. A more realistic raw milk consumption graph might look like Figure 6-6. In this example some day 1 raw milk remained unprocessed at the end of that day's processing. It was consumed first on day 2, then the day 2 raw milk supply was used.

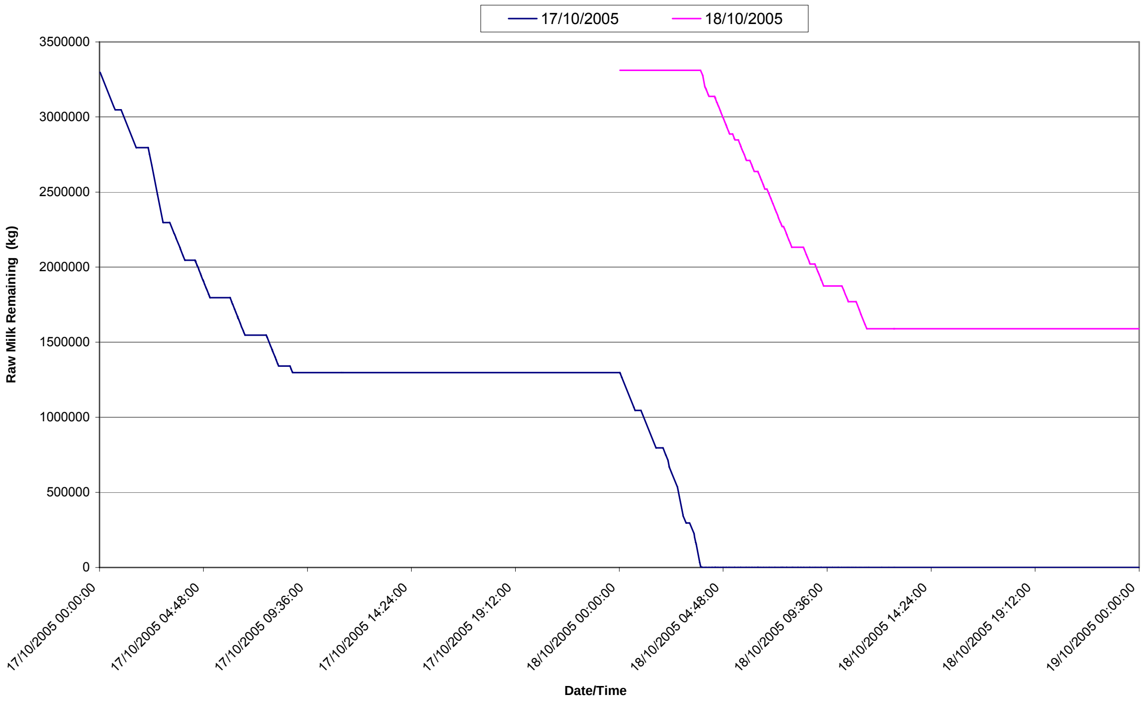


Figure 6-6 –Raw Milk Consumption – 2 Day Simulation – Multiple Processing Facilities

Data for Figure 6-6 was generated using a second process operating simultaneously with the cheese making process. This second process was essentially a sink for raw milk, which was created to consume excess raw milk and generate this data.

6.7 Accumulated Energy Consumption

The pasteurizer model described in section 4.8 was implemented to test the proposed energy transfer mechanism. Though the pasteurizer model was simplified, the test showed the potential of the mechanism. The data generated was used to produce a graph (Figure 6-7) of the total energy consumption by the pasteurizer with a throughput of 35 kg/s, which increased the temperature of the cheese milk by 29 degrees Celsius (from 3 to 32 degrees C).

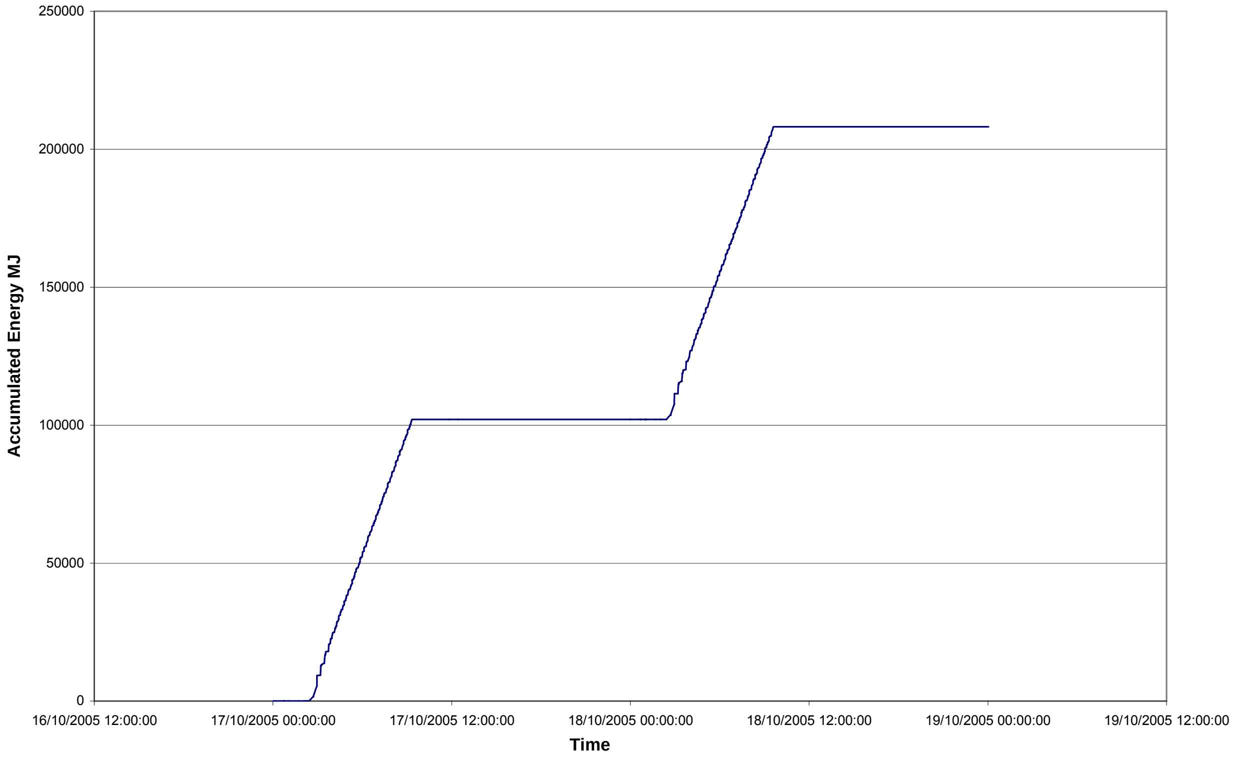


Figure 6-7 – Pasteuriser Accumulated Energy Consumption – 2 Day Simulation

On day 1 of the simulation, 102,132 MJ was consumed by the pasteurizer, while on day 2, 106,017 MJ was consumed.

On day 1 of the simulation 871,030 kg of cheese milk was pasteurised. Using a specific heat capacity for milk of 4000 J/(kg K) this equates to 101,039 MJ of energy required to raise the temperature by 29 degrees C. There is a discrepancy of 1.08% between the calculated result and the simulation's result.

On day 2, 904,530 kg of cheese milk was pasteurised. This equates to 104,925 MJ, a discrepancy of 1.04%.

6.8 Sources of Error

6.8.1 Unsuitable Time Increment

Selecting an unsuitable fixed-interval time increment can result in the model becoming unsolvable. If the time increment used results in an instruction (from a production scenario) to a unit operation being missed, the unit operation will not have the state it is meant to have, when it is meant to have it, for the simulation to run.

For example, if the current time is 08:30:50, and the fixed increment time increment is 30s (meaning the time of the next iteration is 08:31:20), then an instruction for a unit operation to go into (say) FILL QUEUE at 08:31:00 will not be performed with the current software implementation. Therefore, in this work all production scenario instructions are placed on the minute. Consequently the choice of fixed-interval time increments in Table 6-1 are all the whole number quotients of 60 seconds i.e. divided by the numbers

60, 30, 20, 15, 12, 10, 6, 5, 4, 3, 2, 1.

The (arbitrary) selection of the minute as the point at which to place instructions also defines the maximum allowable fixed-interval time increment – i.e. in this case 60 seconds. If the instructions were placed on the ½ minute, the maximum fixed-interval time increment would be 30 seconds.

6.8.2 Fractional Second Error

If the time required to change the state of a unit operation is less than one second, an error will occur that will affect the mass balance. For example, consider a capacitive unit which is emptying into a downstream capacitive unit (Figure 6-8).

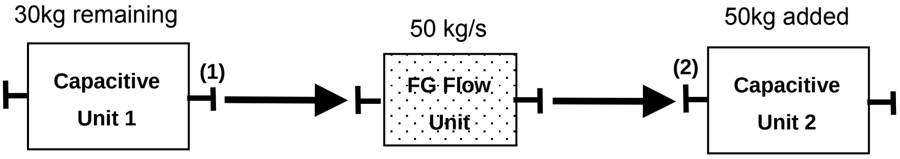


Figure 6-8 – Flow Generating Flow Unit – Capacitive Unit Upstream & Downstream

The capacitive unit's outlet port's flowrate (port 1) is 50kg/s. The capacitive unit has only 30kg remaining, so the time required to empty it is 0.6s. Because the smallest allowable time increment is 1 second, capacitive unit 1 will lose 30kg, but capacitive unit 2 will add 50kg. The overall mass balance will gain 20kg.

7 Discussion

The cheese making model developed here takes an amount of raw material, processes it, and produces a quantity of manufactured product. The model reproduces expected features of the cheese making process and its constituent unit operations, such as the cheese vat batch cycle, and the use of multiple cheese vats to give continuous production. Process and production data is generated which can be used by a variety of decision makers.

The model implementation developed in Visual Basic .NET uses two software technologies, CAPE-OPEN and OOP to:

- construct a software tool for defining unit operation models,
- integrate them into a chemical process model,
- build production modelling scenarios over varying time horizons, and
- run simulations which generate process data.

In this chapter the key features of the model are discussed in the context of the aims of this work – i.e. to develop a multiscale model of cheese production capable of delivering information for operations and management level decision makers.

7.1 Multiscale Model Analysis

Though a cheese making process and production model is presented, the question remains whether the model as implemented is a multiscale model. The key to a multiscale model is the integration of individual partial models which describe phenomena of interest at different time, length and detail scales (section 2.2).

This section will examine:

- whether the individual models are partial models (i.e. models representing phenomena at different time, length and detail scales),
- whether the partial models are integrated and what the integration frameworks are. Can they be classified according to the integration classification scheme discussed in section 2.2.2?

If partial models exist and are integrated the model presented here is a multiscale model.

Table 7-1 - Partial Model Scale Comparison

Partial Model	Phenomena of Interest	Length Scale	Time Scale
Production Model	The control of the all production facilities to process an amount of raw milk and manufacture specified amounts of products at the specified time.	~10^1 – 10^5 metres. The distribution of all the important features of production (i.e. raw milk supply, production facilities).	~10^5 – 10^8 seconds. The duration of the production schedule under consideration.
Raw Milk Supply	The quantity of raw milk available for processing on any day in the year.	~10^3 –10^5 metres. The distance of farms from production facilities.	~10^5 – $10^{7.5}$ seconds. The duration of the milk curve cycle.
Cheese Making Process	The transfer of material and energy into the cheese making process and between unit operations.	~10^1 – 10^2 metres. The physical size and spread of the cheese making plant.	~10^3 – 10^5 seconds. The duration of batches and cycles of unit operations.
Unit Operation (15 of – see 5.1.1)	The possible states of the unit operation and the expected duration of the states.	~ 10^0 – 10^1 metres. The physical size of the unit operation.	10^2 – 10^5 seconds. The time for the unit operation to pass through its different states during processing. Some unit operations, such as cheese vat, cycle in a few hours. Others such as a milk storage silo cycle in 24 hours (i.e. maximum allowable time between cleanings).
Capacitive Unit's Material Content Calculation	The calculation of material quantity in the capacitive unit operation when it is in a material transfer state (i.e. FILLING, EMPTYING, or FILLING / EMPTYING).	~ 10^0 – 10^1 metres. The physical size of the storage capacity of the unit operation.	~ 10^0 – 10^4 seconds. The time that the unit operation exists in a material transfer state.

7.1.1 Implemented Partial Models and Integration Frameworks

From the list of models implemented here (Table 7-1) it can be seen that they represent different time and/or length scales, and meet the criteria to be classified as partial models. In some cases the classification of integrating frameworks is not clear.

7.1.1.1 Production Scale Model

Here, the system domain consists of all the processing plant options (one of which is the cheese making process) and the raw milk supply model. This is a discrete – continuous hybrid model, where an amount of raw material for the day being modelled is input, and continuous amounts of product are output. The production scale model appears to meet the classification criteria of more than one of the integration frameworks proposed by Ingram et al. (2004) and Cameron et al. (2005).

At first glance it appears that the entire system is modelled at the microscale and the results are converted into macroscale variables (i.e. a simultaneous integration). The production macroscale model is comprised of the raw milk and cheese making process partial models, plus the other processing options which are not implemented here (Figure 7-1).

One of the microscale models (i.e. the cheese making model) produces a data set of manufactured final product. This data set is a macroscale model and is a totalised time based series, which is consistent with the definition of a simultaneous integration framework.

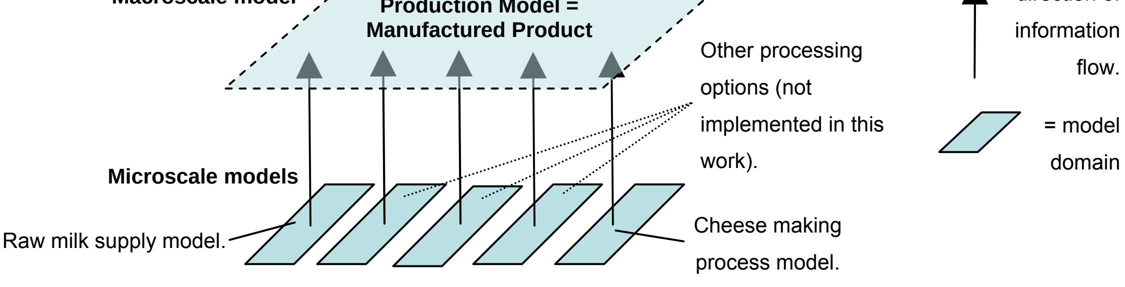


Figure 7-1 –Production Model from Simultaneously Integrated Partial Models

However this framework is defined by Cameron et al. (2005) as having unidirectional information flows only, and therefore does not account for the bi-directional nature of information flows seen here: i.e.

- control of the cheese making process model being achieved by unit operation state instructions being passed to the process model (using the production scenario software class discussed in section 4.3.5).
- production allocating raw milk to each of the processing options by dividing up the total daily raw milk supply.

Another possible integration framework classification which could apply here is multi-domain, where the microscale and macroscale models describe separate but adjoining parts of the whole system (Cameron et al., 2005). Here, the cheese making process model is the macroscale model, and the raw milk curve is the microscale model, with the combination of the two being the production model.

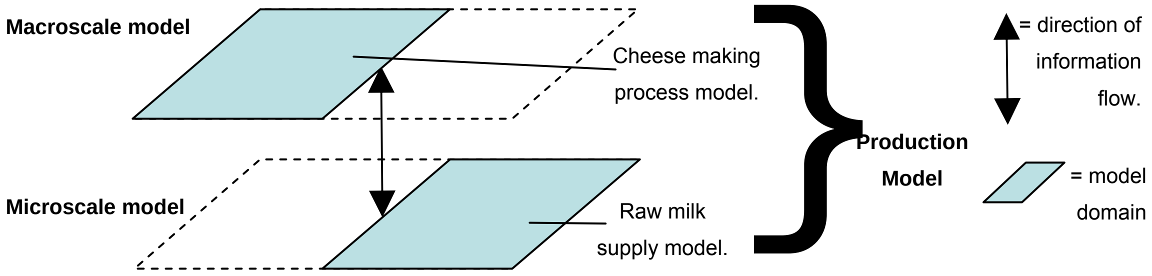


Figure 7-2 –Production Model from Multi-Domain Integrated Partial Models

The multi-domain framework classification does not appear to either satisfactorily describe the interactions between the production model and its component partial models, nor does it incorporate the controlling of the process using unit operation state instructions.

Though other integration frameworks have been defined by Cameron et al. (2005), none fit the production model as well as the simultaneous or multi-domain frameworks. The nature of the integration framework between the models which form the production model remains unresolved. It may be that the frameworks defined by Cameron et al. (2005) do not apply here, or the different partial models (i.e. the cheese making process and the raw milk models) are integrated using different frameworks.

7.1.1.2 Raw Milk Supply Model

The raw milk model provides a boundary condition on the cheese production model for a particular day, by placing an upper limit on the amount of raw milk which is available. Here, the raw milk supply model (microscale) supplies an amount of milk to the cheese production model (macroscale). As discussed in section 4.3.3, an amount of milk is made available to the process on each 1 day production scenario.

In this implementation, the complete yearly milk curve is stored as a database table, and as each 1 day production scenario is iterated, that day's raw milk is added to the Raw Materials software collection class.

7.1.1.3 Cheese Making Process Model

The cheese making process is a sequential modular model, where all the unit operations are connected into a process flow sheet. The model of the cheese making process is the composite of the connected unit operation microscale models (Figure 7-3). The integration framework classification which seems the best fit here is the embedded integration framework, where the microscale unit operation is embedded within the process model.

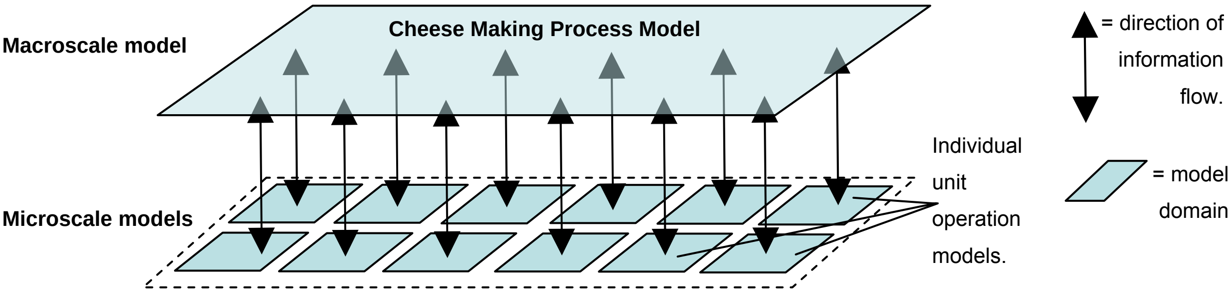

Macroscale model
Cheese Making Process Model
Microscale models
Individual unit operation models.
= direction of information flow.
= model domain

Figure 7-3 – Cheese Making Process Model using Embedded Partial Models

From any single unit operation's perspective, the rest of the cheese making process is the macroscale model while the unit operation itself is the microscale model. In other words, a unit operation does not need to know the state, or any state variables, of any other unit operation. The Process Modelling Environment (section 2.3.1) manages intra-process data, such as connection information and material port flowrates, between the micro- and macro scales.

The connection of unit operations into a process flowsheet performs a single task. It provides a pathway for the movement of material, energy, and information between unit operations. The cheese making model generates data on phenomena such as production, energy and material use and unit operation behaviour at any point in time.

7.1.1.4 Unit Operation Models

Unit operation models are constructed using a combination of empirical data and mechanistic phenomena. As shown in section 4.3.2, the generalized models are a user defined collection of possible states of existence of the unit operation, various rules defining a state's existence and duration, and boundary conditions.

Where a state has a duration, generally the value is set using empirical data. For example, a cheese vat's CUTTING state's duration is set based on the empirical data gathered from previous batches. However, in one case, unit operation models use a partial model to calculate the duration of a state and provide important macroscale model detail.

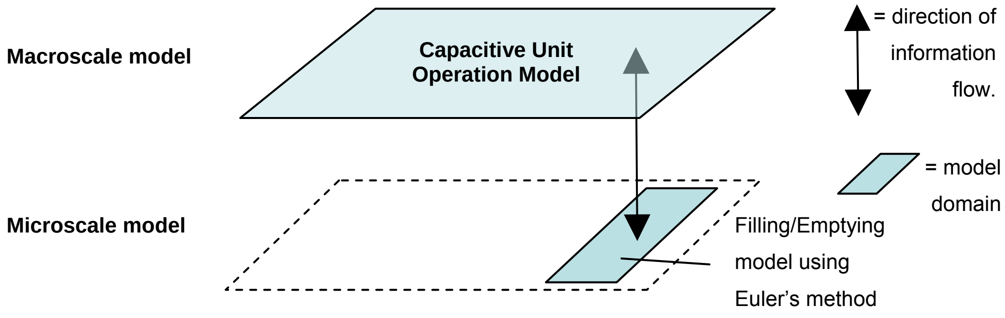

Macroscale model
Capacitive Unit Operation Model
= direction of information flow.
= model domain
Microscale model
Filling/Emptying model using Euler's method

Figure 7-4 – Capacitive Unit Operation Model using an Embedded Partial Model

The filling and emptying of a capacitive unit is modelled using a microscale partial model (see section 7.1.1.5) which provides information on the material content (e.g. the material volume) within the unit operation. The unit operation macroscale model spans the system domain. The filling/emptying model spans only a small part of that domain (Figure 7-4). This is an example of an embedded integration framework.

7.1.1.5 Unit Operation Material Content Model

When the unit operation is a capacitive unit, it has among its possible states FILLING, EMPTYING and FILLING/EMPTYING. When a capacitive unit is in one of these states, and there is flow at one of the material ports, the unit operation's material content is changing (increasing or decreasing). This change is calculated using Euler's method (see section 7.7.2).

7.1.2 Where do the Production Scenarios Fit?

One final issue is where the so called production scenarios defined in section 4.3.5 fit into the multiscale modelling context. As discussed these instructions are central to the control of the process model. They are responsible for turning the process model which would otherwise only be capable of simulating a single 1 day period into a model capable of simulating any time horizon. In other words, they allow the cheese making process scale to be extended in time and used as a cheese production scale model.

Production scenarios do not intuitively appear to be models. They are a set of instructions which tell the process what state a unit operation must be in. The process modelling environment then uses them to set the unit operation state. Perhaps they should be thought of as boundary conditions for the process model. This issue remains unresolved.

7.1.3 Analysis Summary

The model presented here is a multiscale model. Multiple partial models spanning different time and length scales are integrated into a production model capable of simulation a different (and greater) time span than its constituent partial models.

7.2 Multiscale Model Performance

In this section the performance characteristics of the model are examined in the context of the model’s solution speed, data quality (i.e. accuracy and detail), and data quantity. The model was not compared against an actual process, which would be the real test of the model’s accuracy.

7.2.1 Data Quality

Once a solution to the modelling problem is found a simulation can be performed using either of the two time increment modes (i.e. fixed-interval or discontinuity). In the simulations performed here, both the discontinuity and the fixed interval with a 1 second increment generated data of equal accuracy at both the unit operation and process scales. When the fixed interval time increment was increased, accuracy was reduced and detail was lost.

For all simulations the largest overall mass balance error on the cheese making process for a 2 day simulation which processed 2 million kilograms of milk was 1%. The smallest was zero. The primary source of mass balance error occurred when the time increment to the next unit operation state change should have been less than 1 second. However, the software was limited to a minimum time increment of 1 second. To improve the accuracy of the model a mechanism for incrementing fraction of second time increments would need to be implemented.

Performing a simulation with a 1 second time increment is inefficient. Often nothing of significance occurs in the production process’s unit operations over a particular 1 second period (e.g. there is no change in unit operation volumes or states). This results in data being generated which contains no important information, and computer processing and data storage capacity is wasted.

Increasing the size of the time increment reduced the number of unnecessary iterations (therefore reducing both the number of calculations required and the amount of data generated), but resulted in data errors and possible simulation solution failure.

The discontinuity mode time increment mechanism overcame this problem. In this mode a forward calculation is performed to find at what time increment the next unit operation state change will occur. So the model solution will have fewer steps and consequently produce less data.

The model operating in both fixed increment and discontinuity mode captured all the important features of the cheese making process and characteristics of individual unit operations. For example, features such as the continuous – batch nature of the cheese vats were successfully modelled.

The model accurately generated the correct amount of manufactured product based on the mass of final product material (i.e. cheddar cheese) produced.

Overall, the model performed with very low error and accurately modelled the behaviour of unit operations and the overall process on short time horizons (up to 6 days tested). Longer time horizon simulation testing is needed to validate the model on monthly and yearly time horizons.

7.2.2 Solution Speed

The solution speed of the multiscale model depends on several factors:

- The simulation's time horizon (i.e. hours, days, months, or years). Because of the time-interval incrementing mechanism driving this model, the longer the time horizon for the simulation, the longer the solution time.
- Time increment calculation mode. A fixed interval time increment mode simulation solution (note that the time interval chosen must give a solution) will always take longer than the same simulation run in discontinuity mode.
- The type of process. A process which operates continuously with few unit operation state changes will require less computer processing than a batch process where unit operations have many state changes.
- The number of unit operations in the process. For a given simulation timer control time interval, as unit operations are added to the model, the

computer processing required to run the model can be increased.

> A simulation involving one capacitive unit operation, operating a 20ms iteration rate (see section 4.10.4) with a 1 second time increment took 1 minute to simulate 50 minutes. When the number of unit operations was increased to ten, the same iteration rate and time increment took 3min 7s to simulate 50 minutes.

Solution speed is an important factor in the usefulness of a model from an industrial perspective. A simulation that takes days (which is conceivable using this model) may be acceptable if the time horizon being modelled is years, but not so acceptable if the time horizon being modelled is days. Fortunately the discontinuity calculation mechanism facilitates large time steps and shortens the solution speed. However as microscale partial models (at the reaction level) are added it may be that a simulation's maximum time step may become smaller. In this work the maximum time step seen was over 5000s.

Speed can be improved using more powerful computer processing. The 1600MHz Intel Pentium M processor used here limited the model's iteration speed to a maximum speed of about 20ms (depending on the number of unit operations in the simulation) before the processor reached capacity. The smallest possible standard iteration speed available in Visual Basic .NET is 1ms.

It maybe that software changes, such as improving the discontinuity calculation mechanism, will decrease the solution time by reducing the number of calculations. FOR…NEXT software loops for example can slow processing, and there may be gains in efficiency available by re-examining the need for some of these.

7.2.3 Data Quantity

This modelling technique potentially generates enormous amounts of data.

For example, in the cheese manufacturing process modelled here (with 28 unit operations), a simulation operating a 1 second fixed-interval time increment modelling a 6 month period would generate over 15 million records for a single variable (e.g. volume). A 60 second time increment would generate over 260,000 records. If 5 properties are reported on, the simulation's data burden for a 1 second time increment becomes nearly 79 million records (1.3 million records for a 60 second fixed increment).

In the case of a discontinuity mode simulation, the number of data records per property becomes a function of the placement of a production scenario's unit operation state instructions and the number of variables being reported. This mode will always generate less data than the fixed interval mode.

7.2.4 Choosing the Time Increment Mode

The choice of one time increment mode over another depends on the requirement of the user. The fixed increment mode (with a small time increment) is used when designing and testing a new modelling solution. The discontinuity mode is the better alternative when performing the actual simulation once a modelling solution has been found. It should also be useful when performing optimizations on multiple production options (not implemented).

The calculation of the time step to discontinuity for linear and non-linear simulation models is discussed in section 7.7.3.

7.3 Decision Making Information

The data generated by the cheese making process model implemented here has uses in multiple levels from the dairy business, from process operations and production planning, to supply chain management and process design.

7.3.1 Gantt Charts

Gantt charts (see section 6.4) are of interest to production operational and management levels. The information can be used for scheduling and maintenance planning, and process optimization. They give a useful visualization of the behaviour of a combination of unit operations over time relative to each other, and of the utilization of an individual unit operation. For manual processes, they also provide a recipe for plant operators to run the process. For automated processes, they provide the information needed to programme the plant's control system.

Consider a manually operated cheese plant, with a cheese vat batch cycle which generates the Gantt chart shown in Figure 7-5.

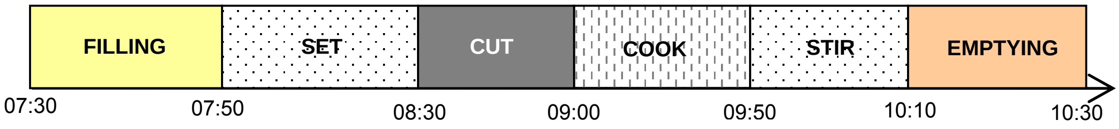


Figure 7-5 –Cheese Vat State Gantt Chart (not to scale)

The operator can use the Gantt chart to operate the cheese vat as follows.

1. The cheese vat should start FILLING at 7:30. The operator will initiate the actions required to achieve this (e.g. opening of cheese vat inlet valve).
2. At 07:50 the unit operation should go into the SET state, and the operator might close the inlet valve (the addition of starter bacteria and rennet is not considered here).
3. At 8:30, the operator sees that the SET state should be complete and the curd should be formed. The operator will check the strength of the gel, and initiate the CUT state (e.g. by switching on the cheese vat's cutting knives).
4. At 09:00 the operator will stop the CUT and initiate the vat's COOK (e.g. by opening a hot water valve which adds hot water to a heating jacket surrounding the vat).
5. At 09:50 the operator will start the STIR by switching on the stirring blades. It may be that the operator will monitor the pH of the curds and whey mixture during the STIR.
6. At 10:10, if the pH is correct, the operator will terminate the STIR, and open the cheese vat's outlet valve to begin EMPTYING the vat.

From the plant operator’s perspective, each state change may require one or more actions. The Gantt chart in this example shows the operator when to begin and end the tasks associated with a particular state.

The combined cheese vat Gantt chart shown in Figure 6-2 gives the plant manager information on the utilization of the cheese vats and redundancy in the process. For example, because the cheese plant must be operated continuously, it is important that there is some redundancy in the cheese vats to (say) allow for a batch which takes longer than usual to complete. From Figure 6-2 it can be seen there is little redundancy in the modelled process. If a batch required a longer cook stage than normal, the vat might not be available for emptying onto the cheese belt when needed (to keep the process continuous). Similarly, the vat might not be ready for filling on schedule. In this case the plant management may decide to add an extra cheese vat to better ensure continuous operation.

Over longer time frames, Gantt charts can be used to examine the availability of the cheese making process over (say) the milk production season. This is particularly useful to management who are involved in production planning. Because there are long periods of under utilization due to the varying milk supply (discussed in section 3.1.2) a Gantt chart of a complete process gives information on the availability of the process at any point in the season. This information would also be used for human resource allocation and plant operator shift scheduling.

Gantt charts are also useful for process design and optimization. For example a proposed process can be modelled using different cleaning and maintenance regimes, capacitive unit operation volumes, and flowrates. The effect of these variables on process and unit operation availability can be analyzed.

7.3.2 Time Series Graphs

In this work several time series graphs are presented:

- unit operation volume time series (section 6.3)

- manufactured product (section 6.5)

- raw material consumption (section 6.6)
- accumulated energy consumption (section 6.7)

Between them they provide important information for plant operations, production planning, maintenance scheduling, inventory control, sales and supply chain, and dairy business management decision makers.

The production manager can analyze the viability of different short, medium and long term manufacturing scenarios and different production schedules using the manufactured product and raw material data.

The dairy business manager can use production information in conjunction with sales and cost models (which also use production data) as part of the overall business plan forecast and analysis.

Manufactured product information is important to production planning decision makers to ensure they can meet manufacturing requirements. This is also important to inventory control.

Raw material consumption information is important to supply chain decision makers to ensure the timely ordering and delivery of raw materials (aside from raw milk) used in the manufacturing process.

This information can be used in the form presented or other forms which may be useful for other types of decision making. For example, whereas the accumulated energy consumption would be useful for cost planning, time series data of actual energy consumption is useful for production planning.

Consider a site which has multiple processing options which compete for a finite electricity supply. It may be that, based on a modelling scenario's actual energy consumption data, production scheduling is reconfigured to allow the competing processes to operate at different times. This type of analysis is also useful if electricity prices fluctuate over the short to medium term. An analysis of the timing of electricity consumption will allow decision-makers to schedule production for times when electricity prices are lower.

7.4 Modelling Unit Operations using their States

Once a unit operation's possible states have been identified (see section 4.3.2) and defined (in sections 4.9 and 5.1) the behaviour of a unit operation can be modelled, and the process controlled. The use of the state as the core process control mechanism which is used to create multiple time period modelling scenarios (by forcing a unit operation into a particular state at a particular time) has been discussed in sections 4.3.5 and 5.4. This section will discuss using a unit operation's state to:

- model unit operation behaviour
- integrate lower scale partial models and calculate process information
- enforce unit operation business rules.

7.4.1 Unit Operation State Behaviour

It is intuitive that an entity's behaviour is dependent on the state it is in. In this work a unit operation's behaviour is driven by its current state. This was achieved by two mechanisms. One mechanism uses software code. The other mechanism uses the state collection of a unit operation.

The unit operation software object is programmed to run certain code in certain states. For example, a pump (i.e. a flow generating flow unit), when in an ON state, sets its own material port flowrates to a predefined non zero value. This is all it does in that state. If the pump's state is OFF, it will set its material port flowrates to zero.

Other unit operation behaviour is dictated by what type of state it is in (i.e. static state or dynamic state). For example a cheese vat's batch process is modelled using dynamic states (see section 4.9.2) which model the various stages of the batch process.

7.4.2 A Unit Operation's State as an Integration Interface

Following on from the previous section, it is proposed here that a particular microscale partial model which is integrated with a unit operation macroscale model will be run when the unit operation is in a particular state. In other words, the unit operation's state is used as the mechanism for deciding which microscale model to recalculate. This is achieved by associating a microscale partial mode with a state. When a unit operation is in a state which has a microscale partial model associated with it, that partial model is recalculated, and process information which is relevant to that state is obtained.

In this work, the model of a capacitive unit operation (at the macroscale) is integrated with a material content microscale model (discussed in section 7.1.1.5). The microscale model will recalculate when certain criteria are met. If an upstream capacitive unit is in a suitable state (i.e. EMPTY QUEUE or EMPTYING), and has a non zero material content, that capacitive unit will begin emptying subject to a downstream capacitive unit being in a suitable state (i.e. FILL QUEUE or FILLING) and having capacity to receive material. There must also be flow at the material ports, and the current global time must be different from the last time the microscale model was recalculated.

The concept of using the state to integrate microscale partial models with unit operation model is developed further in section 7.5.

7.4.3 Using States to Implement Business Rules

Business rules can also be implemented using states. For example if a unit operation must be cleaned every 24 hours, then the unit operation will have a compulsory state (see section 4.9.3) assigned to it. The unit operation will be forced into the CLEANING state if it is not placed manually into that state (by a production scenario) before the maximum time interval between the state occurring has passed.

The compulsory state concept allows other business rules, such as preventative maintenance regimes, to be incorporated into the model.

7.5 Adding Other Scale Partial Models

Integrating a unit operation (e.g. a reactor) macroscale model with microscale partial models will improve the detail of the unit operation model. Integrating this multiscale model with a macroscale model allows that model to access the production data generated by a simulation. In this section a mechanism which allows the integration of unit operation partial models with both macro- and micro- scale partial models is presented.

The decision to integrate will depend on the level of interest in the data generated by any model and its contribution to the task of achieving the overall modelling goal. For example whether a more accurate and detailed representation of the variable(s) supplied by that model will significantly improve the accuracy of the system model. So a microscale partial model could be used in lieu of a constant value to provide more realistic model behaviour.

Using object oriented programming (OOP) software development, the potential to add partial models appears unlimited. Not withstanding processing and data management costs, OOP should allow the incremental addition of partial models without requiring extensive changes to the core software model.

7.5.1 Microscale Partial Models of Unit Operations

It seen in section 7.4 a unit operation’s state property is a useful interface for integrating unit operation microscale partial models.

The capacitive unit software class is currently programmed to perform certain calculations when in the FILLING or EMPTYING state (i.e. its volume recalculation method discussed in section 7.7). In this case the capacitive unit performs the calculation for the duration of its existence in either of these states.

As discussed in section 4.9 some unit operation states have a time component associated with them. Some expire after a predefined lapse of time from when that state is first attained (e.g. COOKING in a cheese vat) and are used to simulate the steps in a batch process. Others must occur at predefined intervals (e.g. CLEANING in most unit operations). Here, the time component is set manually by the user during configuration of the model.

However, in some instances it may be possible to calculate the time component of a state using a microscale partial model integrated with the unit operation partial model. Consider a cheese vat batch. The vat's state collection is used to step through the batch states that represent the stages of reaction and processing that produce the curd.

In Figure 7-6, interface **A** shows a representation of the current implementation (not all states in the state collection are shown). The cheese vat's SET, CUTTING and COOKING state's durations are currently configured by the user. As the cheese vat simulation progresses, and the cheese vat simulation needs duration data on the states, the state collection supplies it.

A more accurate and detailed multiscale model would use a microscale model to calculate the duration of each state based on process conditions. It might work like this.

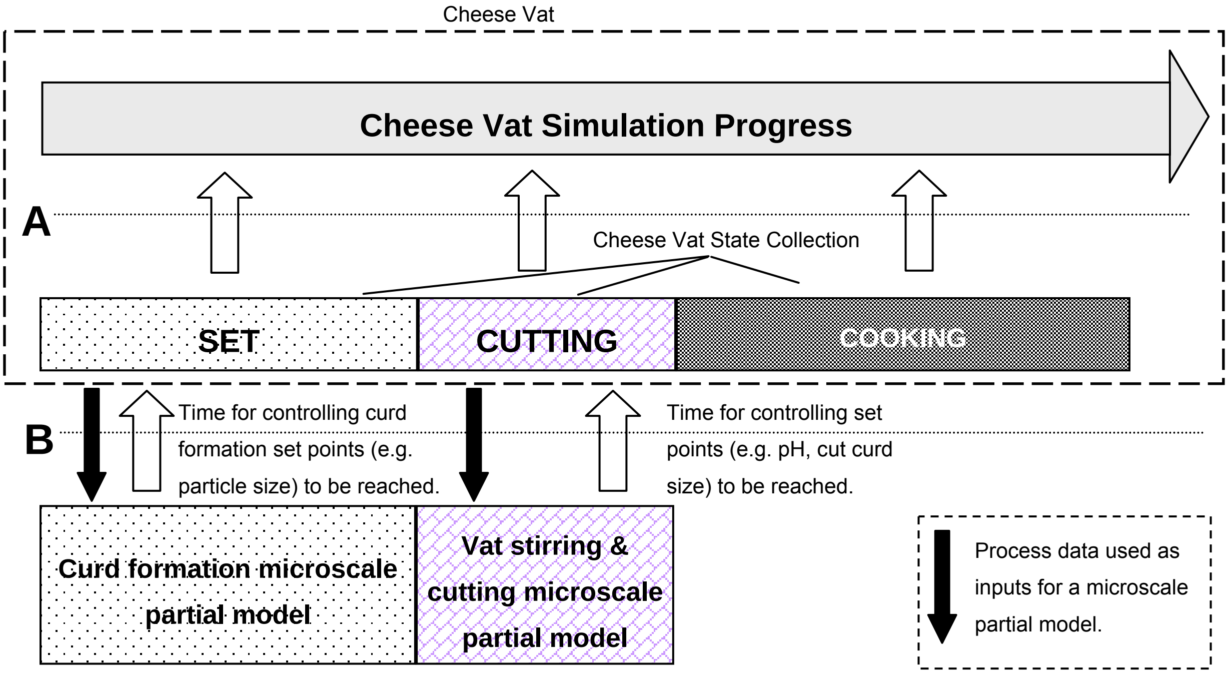

Cheese Vat
Cheese Vat Simulation Progress
A
Cheese Vat State Collection
SET
CUTTING
COOKING
B
Time for controlling curd formation set points (e.g. particle size) to be reached.
Time for controlling set points (e.g. pH, cut curd size) to be reached.
Curd formation microscale partial model
Vat stirring & cutting microscale partial model
Process data used as inputs for a microscale partial model.

Figure 7-6 – Cheese Vat Status Duration Calculation

The cheese vat batch macroscale model is integrated with a curd formation microscale model (Figure 7-6). At the start of the simulation, the duration of the

SET state is calculated. The unit operation passes the curd formation model process information (components, pH, temperature). The curd formation model calculates the length of time for the model to reach the target curd consistency and rigidity, and returns it as the SET state's duration.

Similarly, a partial model could be used provide the CUTTING state's duration by calculating the time for the required average curd size and whey pH to be reached.

Figure 7-6, interface **B** shows the cheese vat's SET and CUTTING state durations are calculated by the partial models. The COOKING duration remains user configured (i.e. a fixed value). The states' duration values are then supplied to the cheese vat model in the currently implemented way (Figure 7-6 interface **A**).

A state's duration may need to be recalculated during simulation. For example when a unit operation's mass balance changes, or if process conditions which are inputs to a microscale partial model change (e.g. pH, temperature, cutting or stirring rate).

The current multiscale model can provide flow, volume, and unit state data. In the above example, process detail comes from a scale below the cheese vat unit operation. Reactor and process control properties such as pH, particle formation, temperature (e.g. for an exothermic reaction) will be provided by the microscale partial models.

7.5.2 A Macroscale Partial Model

A simulation can generate datasets which can be used by other scale models. Consider cheese manufacture. Over time cheese is manufactured and placed in storage (i.e. the Manufactured Product collection) – Figure 7-7 interface **B**.

A sales and marketing partial model that sells cheese would access the Manufactured Product collection (Figure 7-7 interface **A**) to obtain the required amount of cheese. The cheese in the storage varies as manufactured cheese is added and sold cheese is removed.

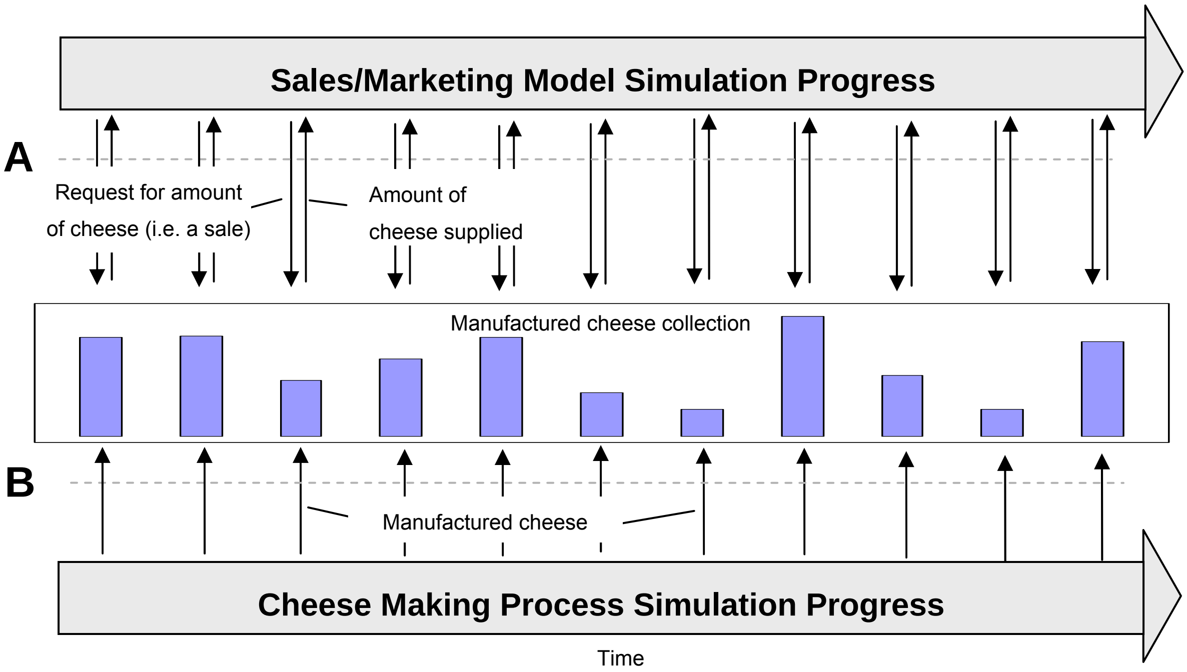

Sales/Marketing Model Simulation Progress
A
Request for amount of cheese (i.e. a sale)
Amount of cheese supplied
Manufactured cheese collection
B
Manufactured cheese
Cheese Making Process Simulation Progress
Time

Figure 7-7 – Cheese Process Manufactured Product Collection

7.5.3 Other Scale Partial Model Possibilities

Other partial model possibilities exist. Wherever an input variable can be calculated or obtained from empirical data, a partial model could potentially be integrated into the multiscale model to provide that variable's value. Possibilities include:

- Raw material and utility cost data which experiences price volatility (such as electricity) could be provided using a partial model. For long term modelling, most costs will not remain constant and it may be desirable to model them from a partial model (e.g. labour and raw materials).
- Dynamic modelling of startup and reactions.
- A variable speed pump, where the flowrate is a function of the power supplied could be connected via an energy port to a flowrate/power supply partial model.

- Pumps could be modelled using pressure-flow models to provide greater process detail such as providing a material object with a pressure property value.
- Pipe work could be modelled using the capacitive unit class with a pressure drop property (which in itself might use a partial model consisting of a Reynolds number calculation).
- Heat transfer in a heat exchanger might use a partial model to calculate the changing heat transfer coefficient due to scale build-up. This in turn would be used to schedule maintenance and model energy demand.

Each additional microscale model added to the multiscale model will increase the data processing and storage demands of simulation.

Possibilities also exist for integrating this multiscale model with other macroscale models other that the sales and marketing model discussed in section 7.5.2. For example, the raw material consumption model could be used as input data for a supply chain management model. The amount and timing of raw material consumption could be used for purchasing and warehousing modelling.

Another possibility is price data associated with production, such as electricity costs and manufactured product value could be used as inputs into the financial control models.

7.6 Incorporating Actual Plant Data

The potential exists to incorporate actual plant data into the model. The data could be historic or real time. This involves taking recorded or live data (e.g. flowrates, material components, temperatures, raw material and utility prices) and using it as the initial or boundary conditions for the relevant partial models. The current implementation will not facilitate this, but its implementation has two benefits.

Firstly it would enable the decision maker to use the model to provide a more accurate prediction of process performance, and raw material and utility consumption. This would allow an immediate update of reported production and cost forecasts.

The second benefit would be to improve the model or the process. The causes of inconsistencies between process and model data can be identified and improvements to either made.

7.7 Stepping Partial Models

From sections 5.5 and 7.1 it is seen that the production model generates cheese production data by stepping of the cheese making process, raw milk supply, unit operation and capacitive unit models.

However, the decision on when to iterate each of these models has important implications for the overall performance of the model (section 7.2). This section will look at an inefficient recalculation regime which was first implemented, and show how it was improved. The general implementation of partial model stepping (which drives model recalculation) is also examined.

7.7.1 Inefficient Recalculating of Unit Operation Partial Models

In the sequential modular cheese making process model presented here, the simplest strategy for recalculating the model (and consequently every constituent unit operation microscale partial model) was to iterate it at 1 second fixed intervals for the duration of the simulation. This was the strategy adopted in the first software implementation.

It quickly became apparent that this approach was impractical because of the potentially long simulation solution times involved, and the large amount of data generated. Both these are the consequence of unnecessary recalculation of partial models. Therefore it was desirable to increase the time increment per iteration, and only recalculate a partial model when necessary.

A mechanism was needed to identify what a unit operation's next state would be, and when the change to that state would occur. The simulation could then be incremented using a time step which was a large as possible, while capturing all the occurrences of interest within the process. The search for the change in state is to look for the next discontinuity in the behaviour of the unit operation.

7.7.2 Time Increment Calculation using Discontinuity

The use of the discontinuity time increment method provides a mechanism to ensure the model is recalculated only when some change of interest occurs. This approach makes use of the linear nature of this system. To illustrate this, consider a simple simulation which models the filling and emptying of a storage silo with a material of constant density.

The silo is empty to begin with. At a user defined time t_1 it begins to fill at constant mass flowrate M_F, and continues until the maximum volume V_{max} is reached at time t_2. A period of time elapses before the silo begins to empty at time t_3 at constant flowrate M_E, until the silo is completely empty at t_4.

The volume time series of this cycle is shown in Figure 7-8.

The discontinuities occur when the simulation's actual time (t_g) equals times t_1, t_2, t_3, and t_4. At each of these times an important event occurs in the silo. The aim of the discontinuity time increment method is to identify when the next important event occurs at any point in the simulation.

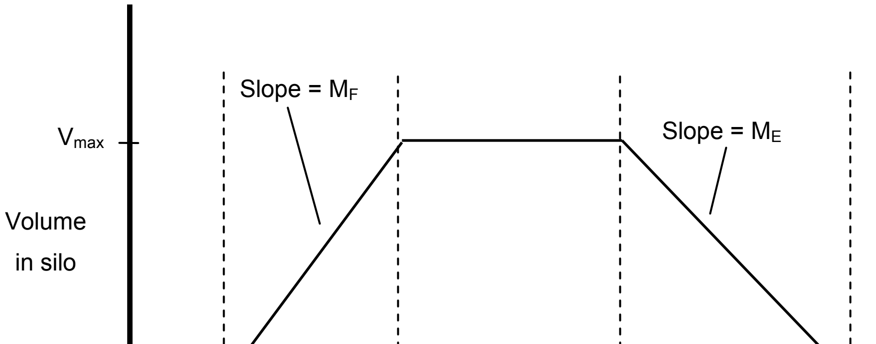


Figure 7-8 – Time Series Graph of a Filling and Emptying Silo Showing Discontinuities

Consider each of the possible situations for t_g:

1. **$t_g < t_1$**

   The silo's state is FILL QUEUE. Because the FILL QUEUE is a static state (section 4.9.1) t_1 will not be identified from an examination of the silo's state collection (section 4.3.2) or those in the production scenario's state collection (section 4.3.5). t_1 will be dictated by the time that the silo's feed pump's state is changed to ON. However, this time point will still be modelled because the simulation will identify that pump's state changes also.

2. **$t_1 < t_g < t_2$**

   The silo's state is FILLING. The t_2 discontinuity point is calculated using Euler's method:

   $\Delta V = M \times \rho \times \Delta t$ where

   ΔV = silo volume change (i.e. $V_{max} - V_{current}$),
   M = the inlet ports mass flow rate,
   ρ = material density,
   Δt = time increment (i.e. $t_2 - t_g$).

with appropriate units to ensure dimensional consistency. All variable quantities except for t_2 are known so the equation can be solved for t_2.

3. **$t_2 < t_g < t_3$**

   The silo's state is EMPTY QUEUE. As in case 1 above, because the EMPTY QUEUE is a static state t_3 will not be identified from examination of the silo's states, but will be dictated by the time that the silo's empty pump's state is changed to ON.

4. **$t_3 < t_g < t_4$**

   The silo's state is EMPTYING. As in case 2 above, t_4 will by found by Euler's method.

The discontinuities in this example can be seen visually by the changes in slope, which imply a change of state, of the volume time series (Figure 7-8). This is not always the case. In this work, more often than not, the discontinuity occurs through some change in state which is not the result of a change in volume (such as the silo changing from the RINSE state to the FILL QUEUE state).

Extending this concept out to a process consisting of multiple unit operation partial models, each unit operation will have its own time increment to discontinuity. The time increment used for the next iteration of the simulation will be the smallest time increment to discontinuity of all the unit operations.

7.7.3 Linear and Non-Linear Simulation Discontinuities

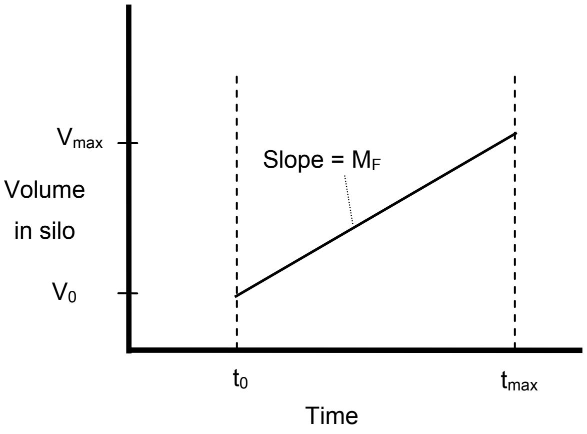


Figure 7-9 – Time Step Calculation – Linear Algebraic

In this work only linear simulation discontinuities are considered. Consider a storage silo being filled by a pump at constant flowrate $M_{F.}$ The volume in the silo is represented by Figure 7-9. Discontinuity occurs when the material volume reaches the maximum volume V_{max}.

Preston and Berzins (1991) describe two types of discontinuity, explicit and implicit. An explicit discontinuity is one where the time to discontinuity is known *a priori*. The volume model above is an example of an explicit discontinuity. The time to discontinuity is found from the linear algebraic equation:

$\Delta V = M_F \, \rho \, \Delta t$ where

ΔV = volume change to full (i.e. $V_{max} - V_0$)

M_F = the inlet port's mass flow rate

ρ = material density

Δt = time increment to discontinuity (i.e. $t_{max} - t_0$)

and can be solved explicitly for Δt.

$$\Delta t = \frac{\Delta V}{M_F \, \rho}$$

V_{max} is the maximum volume, V_0 the current volume, and along with M_F and ρ are known by the simulation. So the time increment Δt can be found.

In an implicit discontinuity not only is the time to discontinuity not able to be obtained explicitly, the final state (e.g. in this example the final volume) might not be known. If the final state is known this is defined as *partially*-implicit. If the final state is not known this is defined as *fully*-implicit.

An implicit discontinuity takes the form *f(x) = 0*.

Consider the tank emptying in a non-linear manner which has a minimum volume set point (V_{min}). Say the volume in the tank is described by the non-linear ordinary differential equation (ODE)

$$\frac{dV}{dt} = k\sqrt{V} \, .$$

and represented in Figure 7-10.

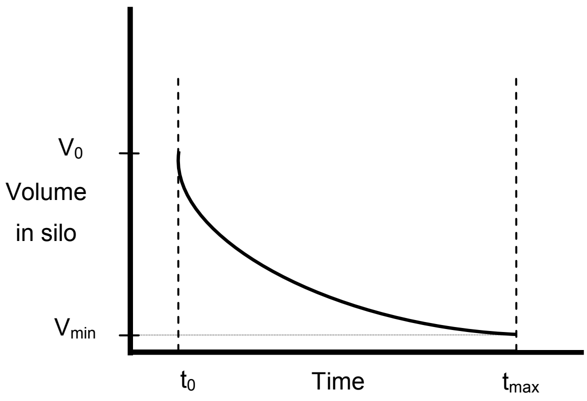


Figure 7-10 –Non Linear Tank Empting

Backward differential formulae (BDFs) and Runge-Kutta numerical methods are used to solve such non-linear ode's (Akai, 1994). Some of these methods (e.g., ODE15S in Matlab, The Mathworks, Inc, MA, USA) allow the specification of implicit discontinuities so that a series of time steps will reach the discontinuity V_{min} efficiently. Because the final state (i.e. V_{min}) is known this is an example of a partially-implicit discontinuity.

7.7.4 Implementing Partial Model Recalculation

As discussed in section 7.2, an important issue from the perspective of model performance when integrating partial models is when to recalculate any particular partial model. Potentially, each partial model in a multiscale could be recalculated whenever the macroscale model it is integrated with is recalculated.

However as shown above this type of recalculation regime leads to computer processing inefficiency. To achieve optimal processing efficiency, it is desirable to recalculate a partial model only when there is a need to refresh the data which the model provides.

The production model is iterated whenever the simulation time reaches 00:00:00 hours (i.e. on a one day cycle). At each iteration:

- the raw milk partial model is recalculated to generate new raw milk supply data for the day (based on the new date), and
- a production scenario for the date is obtained (remembering a production scenario is a set of user defined unit operation state instructions – section 5.4).

The decision to recalculate these models at 00:00:00 is arbitrary, albeit made for the reason that raw milk supply changes on a daily basis. There is no reason for example that the raw milk supply could not be modelled on an hourly basis, to model the movement of milk tankers (e.g. arrival, emptying, cleaning) in the site's milk reception facility.

The cheese making process model then proceeds to iterate multiple times and generate production data. The number of process model iterations is determined by the time increment mode used (i.e. fixed increment or discontinuity). The discontinuity mode of operation is the most efficient because the cheese making process model will only recalculate when a unit operation state change occurs – that is when some time point of interest in the simulation is reached.

In the software implementation here each iteration of the cheese making process iterates every constituent unit operation partial model of the process model. The material content partial model, which is integrated with the capacitive unit operation model, will iterate when two criteria are fulfilled. i.e.:

1. there is flow at any of the capacitive unit's material ports, and
2. the current simulation global date-time is later than the date-time when it was previously recalculated.

In general, the best iteration regime would be one which that only recalculates a partial model when an input variable to that model changes. Here, time dependant models, such as the capacitive unit material content model, are responsible for much of the computer processing demand.

The instances where a partial model would require recalculation are presented.

<u>Example 1 – Mass Balance</u>
Consider a mass balance partial model which calculates the components and flowrates from a cream separator. In this case the model would be recalculated when a flowrate or component concentration changes at one of the input ports. It would also be recalculated when the spjjecification of material from the separator is changed by the user.

<u>Example 2 – Unit Operation State Duration Calculation</u>
Consider a cheese vat's SET state duration calculation (not implemented here). The curd formation SET time in the cheese cooking vat is a function of the concentration of protein in the milk, the rennet concentration in the milk/rennet/starter mixture, the temperature of the milk, and the pH of the mixture

(O’Callaghan and O’Donnell, 1998). When the value of any of these factors change it will be necessary to recalculate the SET time.

Example 3 – Heat Balance

Consider the case of a heat exchanger which uses a partial model to calculate the temperature change of a process side fluid. For a given heat exchanger with constant hot and cold side fluids, and a given desired process outlet temperature, the factors which might change the performance of the heat exchanger include a change in the flowrates, inlet temperatures, and heat transfer coefficients due to buildup of material on surfaces. Partial model implementations should provide a mechanism for deciding when a recalculation needs to be performed based on pre-defined criteria.

7.8 Software Development

This work shows the benefits of taking an object oriented approach to the construction of a multiscale model.

The model is constructed using object oriented programming (OOP) methods in conjunction with technologies such as the Visual Basic .NET software development environment and CAPE-OPEN (discussed in section 7.9). These technologies lent themselves well to the construction of the cheese making process model. OOP classes facilitated the rapid and flexible construction of multiple unit operation partial models while CAPE-OPEN provided technology for the successful integration of, and communication between, unit operation models.

Any chemical process industry multiscale model, by its very nature, consists of extensive data and functionality requirements. The core material, energy, processing, production, and cost information streams alone have multiple sources and multiple interactions. OOP reduces the complexity of the software implementation and allows functionality to be incrementally added to existing models more readily that tradition software programming methods.

For example, even though this model doesn’t include comprehensive energy and costs, there are practical options available for incorporating them which requires the modification of existing (and possibly the addition of new) software classes. Their implementation should not affect the current implementation of material streams, though it could add complexity to the classes involved. The point is that the core model structure remains unchanged.

Another benefit of taking an object oriented approach to multiscale modelling approach to business modelling is that maintenance of the system model from a software development perspective is simplified. If a particular partial model requires changes, which could be as straightforward as new boundary conditions or as complex as a new mathematical model, as long as the inputs and outputs remain the same the partial model can be readily modified without expensive changes to other parts of the software. If the inputs or outputs change modification becomes more complex.

Though the individual models are not implemented as standalone (e.g. dynamic link library) software components, the basic class structure is in place to make the transformation into a distributed application.

7.9 CAPE-OPEN

The CAPE-OPEN documentation is extensive, currently consisting of over 35 separate documents and specifications. Much of this is aimed at the experienced software developer in the form of detailed specifications for constructing process modelling components and environments. They include unit operations, thermodynamics and physical properties packages, numerical solvers, sequential modular flowsheet simulator interfaces, and planning and scheduling tools.

The specification’s emphasise on the oil, gas and refining industry reflects the make-up of the majority of the CO-LaN (CAPE-OPEN Laboratories Network) consortium partners who publish it. However this emphasis does not detract from the usefulness of CAPE-OPEN to software developed for the dairy industry. Many of the concepts are generic to the chemical process industry and the documentation provides useful in-sites into the latest thinking in the area of process modelling in general.

The CAPE-OPEN specification is central to the development of the software used to construct the cheese production model. It was used for, among other things, the construction of unit operation models and their interconnection into process flow sheets (such as ports, materials, port connections).

The models built here are not CAPE-OPEN compliant. However the basic structure is in place allow compliance with the CAPE-OPEN specification to be implemented. CAPE-OPEN compliance would allow process modelling components and environments to be integrated with CAPE-OPEN compliant third-party software.

7.10 Ports

Ports are the mechanism which facilitate the exchange of information (i.e. material flows, energy flows, and other types of information) between unit operations. In this work material ports were implemented, and one energy port implementation was tested. Ports proved simple to implement, and are intuitive to the chemical engineer because they reflect the real world connection mechanism of unit operations in a processing facility.

In this work, multiple outlet material port unit operations have a flow fraction assigned to each material port when the material port is added to the flow unit. This flow factor is based on empirical data (such as the experience of the separation achieved by a cream separator at a particular flowrate).

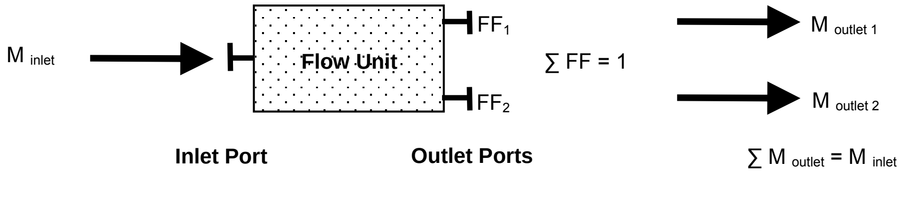


Figure 7-11 – Multiple Outlet Port Flow Fractions

This forces the total inlet flowrate (whether from 1 or more inlet ports) to split according to the fractions assigned. The sum of the flow fractions = 1 (Figure 7-11). Splitting flow using this method is crude. A better way of assigning outlet port flowrates would be to calculate them using a mass balance.

The outlet material ports may have different material objects assigned. So the cream separator implemented here has two outlet material ports. Skim milk is assigned to one outlet port, and cream assigned to the other. In reality a third outlet port could be added to incorporate the cream separator's purge.

7.11 Material Streams

The Material Port – Material class mechanism for modelling material streams (i.e. filling and empting of vessels, material transfer between unit operations, mixing and separation) and connecting unit operations successfully enabled the construction and simulation of a cheese making process model.

Capacitive unit and flow unit classes are used to manage the storage and transfer of material throughout the process and provide core unit operation behaviour, functionality and properties for material transfer.

While capacitive units in practice can have multiple inlet and outlet material streams, here they are implemented with no more than one of each. Stream mixing and separation is modelled using flow units with multiple inlet and outlet ports. In this implementation, for material transfer to proceed, some configuration and operating rules were defined:

- A capacitive unit must be available for filling or emptying before material can be transferred to or from it. It must have available capacity or existing material, and be in a state to receive or release material (i.e. state equals FILL QUEUE or FILLING or state equals = EMPTY QUEUE or EMPTYING).

- Only a flow-generating flow unit can initiate material transfer (i.e. flow) between unit operations.

- The flow-generating flow unit's inlet port must be connected to a capacitive unit's outlet port (i.e. they can only generate flow when immediately downstream of a capacitive unit).

- The flow-generating flow unit must feed into a capacitive unit downstream, though it doesn't have to be connected directly to the capacitive unit (e.g. a pump – pasteurizer – cheese vat configuration).

- A multiple outlet port flow unit (e.g. a separator) must be connected at each of its outlet ports to downstream capacitive units which are available for filling.

- A multiple inlet port flow unit (e.g. a mixer) must be connected at each of its inlet ports to upstream capacitive units which are available for emptying.

- Non-flow-generating flow unit's such as heat exchangers can only be connected downstream of flow-generating flow unit's such as pumps.

- Capacitive units are only connected to flow units. Never directly to another capacitive unit. At the very least a flow unit must separate them for material transfer to occur.

- A capacitive unit can be downstream of either a flow-generating flow unit or a non-flow-generating flow unit. It does not know the difference.

A failed attempt to transfer material may or may not affect the state of the unit operations involved. For example:

- Capacitive units remain unaffected by a failed attempt to transfer material to them. An attempt to add material to a full storage silo with an EMPTY QUEUE state will not change the capacitive unit's state.

- A flow unit that attempts to transfer material (i.e. in the ON state) from or to a unit operation that is not available to give or receive material will be forced into the OFF state.
- Flow units feeding or emptying an available capacitive unit will be in the ON state. Once the capacitive unit reaches full or empty, the state of the capacitive unit will change (from FILLING or EMPTYING) to the next state in the state collection sequence. The flow units will not be able to transfer material and will be forced into the OFF state.

Some of these rules are the result of real world behaviour, others came about because of software implementation constraints and could be made redundant by further software development. For example requiring a capacitive unit to be available for filling or emptying before material can be transferred to or from is a real world constraint. On the other hand requiring a flow generating flow unit to always be upstream of a non-flow generating flow unit may not always reflect real world behaviour and improvements to the software would remove the need for this rule.

7.12 Energy Streams

The energy transfer trial discussed in sections 4.8 and 6.7 was successful, and showed that energy transfer requirements can be managed using the Energy Port – Energy objects.

In the trial, two assumptions are made:

1. energy into (or out of) a unit operation raises (or lowers) the temperature of the material attained from inlet material streams and any material already contained within the unit operation (i.e. in the case of a capacitive unit).
2. all material streams leaving the unit operation are at the same temperature. All outlet port’s materials are at the same temperature. In

practice it is possible for a unit operation to have different outlet stream temperatures. For example a continuous distillation column with reflux. These models could be implemented using combinations of capacitive units and flow units discussed in section 4.6.

An energy balance is required to calculate the outlet temperature. The energy balance includes the energy from all inlet and outlet material ports, all inlet and outlet energy ports, plus, in the case of a capacitive unit, the energy in the material contained within it.

The temperature in a capacitive unit operation modelling a reactor can also be increased if an exothermic reaction occurs. One possible method for modelling this is to use an inlet energy port connected to a reaction partial model to provide the energy generated by the reaction.

There was an error of 1.04% and 1.08% between the model's energy consumption and the calculated energy consumption on days 1 and 2 respectively of the simulation. Because the energy implementation was not done as carefully as the mass transfer implementation it may be this is due to a software bug. However, the accuracy is such that the concept has been proven to be worthwhile pursuing.

More work is needed to implement the many different energy transfer possibilities found in the dairy industry.

7.13 Usability of the Modelling Approach

The question of the usability of this modelling approach is a matter of how easily a model can be defined, constructed, configured, and solved, and whether data from a simulation can be readily accessed and used.

As discussed in Cameron et al. (2005) the traditional approach to the construction of production process modelling involves the construction of a set of equations using balance volume conservation (e.g. mass, momentum and energy balances), boundary conditions, and initial conditions. The equations are then solved using various mathematical techniques. The development of these models is the domain of highly specialized technicians.

A process modelling environment is presented which is used to build the model. That is:

- define unit operation templates,
- construct unit operations from the templates,
- connect them into a process flow sheet, and
- create the production modelling scenarios used to control the simulation.

To utilize this modelling environment a user requires knowledge of the process flow sheet being modelled (e.g. how the unit operations are connected and relevant process conditions such as flowrates) and an understanding of the possible states of unit operations. The ability to apply this modelling technique is within the capability of any plant or process engineer. Specialized modelling expertise is not required.

This modelling approach has benefits to both the user and the software programmer. A sequential modular process model in the form of a flow sheet is intuitive for the chemical engineer user while for the software developer, the model can be easily extended (as discussed in section 7.8).

7.14 Implemented Model Limitations

The cheese making process multiscale model as implemented is limited in several ways.

7.14.1 Batch Modelling

Currently only simple unit operation behaviour is modelled. For example, the cheese vat batch is modelled using a sequence of time-based state changes, which model the steps in the batch (i.e. SET, CUT, COOK, STIR). Material component mass fractions and process conditions which change during the batch are not modelled.

A more sophisticated cheese vat simulation would model, for example, the curd formation in the vat, and the pH development as the batch reaches completion. Thus, the consequence of varying the batch operating conditions could be modelled, with the curd formation model providing material stream property information (e.g. pH) and component data (i.e. mass fractions). An interface for integrating reaction scale partial models is proposed (section 7.4).

7.14.2 Unit Operation Connection Rules

As discussed in section 7.11, the implementation here only allows material transfer when certain flow unit – capacitive unit connection rules are followed. These rules should be extended. For example currently a flow generating flow unit (e.g. a pump) must be connected immediately downstream of a capacitive unit, though flow is not necessarily generated in this manner.

Consider the case of the block forming tower in a real cheese making process. The Block Forming Tower operates under a vacuum. The curd is sucked from the cheese belt to the block forming towers. Here, this situation is modelled using a flow generating flow unit between the Cheese Belt and the Block Forming Tower (Figure 4-16). In reality the Vacuum Pump is downstream of the Block Forming Tower (Figure 7-12).

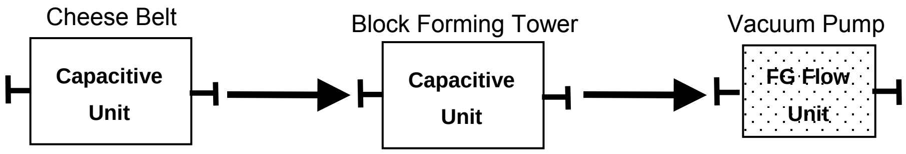


Figure 7-12 – Block Forming Tower – 'Real' Unit Operation Configuration

Currently, a model of this configuration would not generate flow between the Cheese Belt capacitive unit and the Block Forming Tower capacitive unit. Two alternatives to model this situation exist:

- implement imaginary unit operations as discussed in section 4.7

- add functionality to the flow unit and capacitive unit classes to allow the actual situation to be modelled.

8 Future Work

Though a start has been made on developing tools for the creation of models (e.g. CAPE-OPEN) and general theories on multiscale modelling (e.g. integration frameworks), there is much scope for further work in the area of multiscale modelling applied to dairy industry process modelling.

For example, the literature review did not turn up any previous multiscale modelling work examining the process – unit operation scales covered here. Also, the previous work appears to have considered no more than two partial models at adjacent scales. Here, four partial models have been integrated to create a cheese making production model, and the overall model covers 5 time or distance scales. Finally, no previous work was found which examined the role of object oriented programming concepts in multiscale modelling.

So in the general multiscale modelling context, possible areas of future work include:

- multiscale modelling applied to the construction of a process model from unit operation partial models
- multiscale model construction from more than two partial models across multiple scales
- object oriented programming and multiscale modelling

The remainder of this section will examine the possible areas of future work on the multiscale model proposed here. The various areas of work fall into the following broad categories:

- adding functionality to existing classes
- adding partial models
- adding data reporting capability

- making the software CAPE-OPEN compliant
- improving the usability of the software
- adding optimization functionality to the software
- implementing more sophisticated time stepping algorithms
- an alternative modelling goal

8.1 Adding Functionality to Existing Classes

The connection alternatives of the flow unit and capacitive unit classes is currently limited to those discussed in section 7.11. For example no work has been done to examine the suitability of the software for incorporating recycle streams. An analysis of the various unit operation connection alternatives is needed to add flexibility to the modelling software.

The error caused by the inability of a simulation to increment fractional seconds has been discussed (section 6.8.2). If functionality was added which would allow the model to increment by less than 1 second, this source of error would be eliminated.

8.2 Addition of Partial Models

Several possible partial models have been presented in this work.

- A model to calculate the duration of a cheese vat's SET state's existence (section 7.4).
- A sales model to utilise the data generated by the production model (section 7.4).
- Cost models for volatile cost contributing inputs (section 7.4).
- Modelling pipe work (section 7.4).
- Pump models generating energy, pressure and flow data (section 7.4).

- Scale build-up in a heat exchanger (section 7.4).
- A mass balance model to calculate the components and flowrates from a cream separator (section 7.6).
- A energy balance model to calculate temperatures and energy consumption (section 7.6).

Each of these models can potentially be integrated with the multiscale model presented in this work. However, whether any particular model will bring real benefits to the multiscale model is not determined. Some, such as volatile cost models, would at first glance appear to be beneficial to the performance of the model. Others, such as pump models and scale build-up in heat exchangers, might be of academic interest, but their contribution to improving the performance of the multiscale model is less clear.

This leads to another possible area of future work. Currently, there are no clear rules for determining whether a partial model should be added to a multiscale model. Factors such as the importance of the data provided by a partial model, partial model contribution to the modelling goal, cost of implementation (e.g. software development), and cost of implemented operation (e.g. computer processing, effect on solvability) will all be important. However, it would be useful to have some definitive guidelines on when implementing a partial model integration is of real benefit.

8.3 Data Reporting Capability

The data reporting capability can be extended in several ways by implementing scheduling and resource reporting tools. In this work, data was stored in a Microsoft Access database, then imported into a Microsoft Excel spreadsheet where it was converted into the graphs presented in this work. This is cumbersome and inefficient for the user. Two possibilities for improvement are the implementation of a ‘real time’ charting, where data could be generated as the simulation runs, and for graphing tools to be incorporated in the software that would enable graphs and reports to be generated at the completion of a simulation from within the software application.

Tools such as Crystal Reports may be suitable for some of these tasks, especially time series data. A search for a specialized Gantt charting tool was unsuccessful, and it may be that something suitable would need to be developed specifically for this application.

8.3.1 Financial Data

One important reporting requirement which was not implemented here is financial data. The ability to generate financial data generation is a desirable function of a dairy process model. Financial data falls into two categories, production costs and manufactured product value.

The full analysis of the requirements of a financial implementation has not been done, but there are two possibilities.

8.3.1.1 Totalized Financial Data

Raw material object classes, energy, and manufactured product classes could be given a *Value* property, which could then be used to calculate the value of the totalized raw materials, energy consumption and manufactured product at the completion of a simulation.

8.3.1.2 Unit Operation Cost Data

It may be that in some cases it is desirable to analyze costs on a unit operation basis. One approach seems worth considering. This approach involves the creation of a cost software class. In this approach, Material, Energy and unit operation State software classes can have a Cost class attached to them. The cost class has properties, such as value and value units which enable material or energy stream cost data to be calculated.

As material is transferred from unit operation to unit operation as it moves through the process, an outlet material accumulates costs which reflects the additional price of processing within a unit operation. Here, energy and unit operation state costs are contributors to material costs; that is a unit operation's outlet material's costs are functions of the inlet material costs, plus any costs added while the unit operation is in a particular unit state, plus inlet energy costs (Figure 8-1).

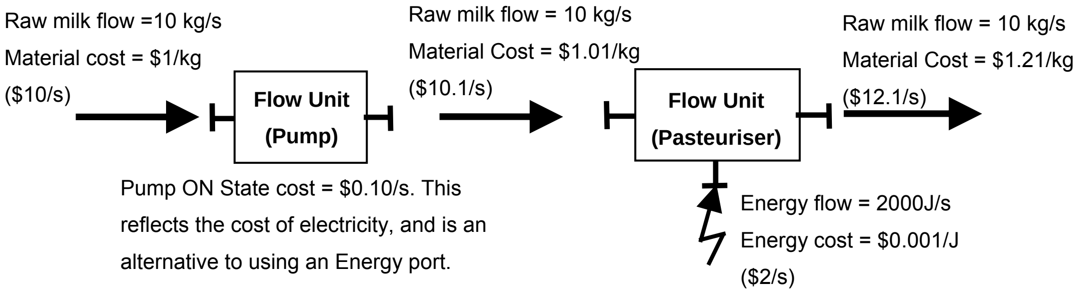


Figure 8-1 – Costs Added by Unit Operation and Energy

Situations requiring cost removal from a material are not considered. Cost removal from a material would imply that the cost of processing the material is reduced by having passed through a unit operation. No situation where this might occur is foreseen.

The cost value of a manufactured product material is calculated during the process simulation, as are the cost values of unit operation outlet materials. Raw material, energy, and unit operation state costs are input by the user, using data obtained from either a fixed value or a data set (depending on the expected volatility of a cost over the simulation's time horizon). For example, a 3 month simulation might use a fixed value for the price of electricity, while a 2 year simulation uses an electricity pricing model which is a function of the time of the year.

Though this approach to implementing costs has not been tested, it seems likely that a cost implementation will involve a Cost class in some form because of the flexibility and programming benefits of classes (section 2.4).

8.4 CAPE-OPEN Compliance

There are benefits in developing CAPE-OPEN compliant simulation environments and unit operation models.

Using a chemical process industry standard specification reduces the resources required to develop the core software functionality. Efforts can be better spent on integrating partial models and developing industry specific functionality.

Also, the potential exists to integrate with third-party CAPE-OPEN compliant process modelling components or environments.

Not all CAPE-OPEN functionality (such as the information port, numerical solvers, physical properties, thermodynamics) was implemented here, and there are opportunities to do so using applications from the dairy industry.

8.5 Improving Software Usability

The software can be made more usable. Tools to simplify the unit operation creation and connection process will both improve the model construction process. For example:

- implementing unit operation type classes (using inheritance),
- object creation and editing wizards,
- drag-and-drop data transfer between object classes,
- visualization of the physical connections between unit operations,
- more realistic graphical representations of particular unit operation types in the process flow sheet.
- A timeline control with the ability to drag and drop unit operations onto date-time positions and have their state set would reduce the time to construct and edit production scenarios.

The implementation of inheritance warrants closer examination because of the benefits it would bring to the user and the developer.

8.5.1 Object Oriented Programming Inheritance

One feature on object oriented programming not implemented here was inheritance. While it is a powerful tool that allows more efficient reuse of code, the complexities and pitfalls associated with inheritance mean its implementation fell outside the scope of this work. However one potential implementation is identified.

Using the “is a” rule for determining when to use inheritance (Pattison, 2001), it is apparent that:

- a Capacitive Unit is a Process Unit,
- a Flow Unit is a Process Unit.

The potential exists to implement inheritance by allowing the Capacitive Unit and Flow Unit classes to inherit Process Unit properties. Furthermore, those classes can be used as the basis for the creation of industry specific unit operation type classes, each inheriting core functionality provided by flow or capacitive unit classes.

For example, a dairy industry specific application would consist of unique unit operation classes used in this industry, such as cheese vats, spray dryers, cooling tunnels, cheese belts, centrifugal separators and block formers for example. Also available would be generic unit operations such as pumps, heat exchangers, manifolds, storage silos.

The flow sheet would be constructed from actual unit operations rather than combinations of capacitive and flow units as is the case in this work. This is a more intuitive approach for the industry user.

8.6 Optimization

Both plant level and business level decision-makers have optimization problems. Naysmith and Douglas (1995) give a comprehensive review of optimization in the chemical process industry and look at the constituent components and tasks required of an optimizer. A general objective function to be maximized is given as:

Objective = Product value - feed costs - utility costs + other variable economic effects.

At the plant level, the optimization problem is focused on maximizing throughput, product quality and product yields while fulfilling the business rules (such as the hygiene requirements) demanded of the dairy industry. A cheese plant manager may for example want to alter operating conditions in a cheese vat based on the component mix of the feed milk to maximize yield, or change unit operation cleaning regimes to increase throughput.

At the production level, the optimization problem may be more complex. Multiple processing plant alternatives mean that, along with the optimization of individual production plants, the objective function may include factors such as co-products (e.g. cream, whey), market influences, product shelf life (i.e. production timing) and product warehousing capability. It may be that the multi-plant production optimum has individual processes operating at sub-optimum conditions to manage the competing demands of different parts of the business.

For example, say the dairy manufacturer can produce cheese or skim milk powder (SMP), and has an order for a quantity of cheese. SMP is manufactured with excess raw milk supply if desired. The optimization problem is to maximize profit over a specified time period whilst meeting the cheese order. Cream, which is a co-product of both the cheese and SMP processes, is itself a product, and also an intermediate material in the manufacture of butter. Once the minimum cheese production is reached the excess raw milk can be used to manufacture more cheese or SMP. It maybe that the cheese plant would operate closer to optimal with greater through-put. But this might be countered by the remaining raw milk (once optimum cheese production is obtained) not being enough to operate the spray dryer.

So the optimal solution lies in both plants operating at sub-optimal conditions.

Though this work did not consider optimization, it is an integral part of the dairy industry modelling problem. Taking a sequential modular multiscale approach may be advantageous because it would allow the solution to be implemented in an incremental manner, something which may not be so easily done using an equation oriented approach to modelling.

Optimization of object oriented models in the multiscale modelling context is possibly a new field of research.

8.7 Implementing more Sophisticated Time Stepping

As discussed in section 7.7.3, this work considers only linear simulation models which can be solved explicitly for the time step to discontinuity. In engineering there are many instances where the model is non-linear and time steps cannot be found explicitly. Models of this type require more sophisticated solution tools using numerical methods such as Runge-Kutta or Backward Differential Formulae methods.

Two tasks are required; the identification of applications which require non-linear models (e.g. emptying of a tank using gravity), and the software implementation of solution methods.

In terms of software implementation, two possibilities are identified. One is to implement these tools within the framework of the software developed. The second is to incorporate third party software tools which provide these solution tools. The second alternative is the more attractive because software already exists (such as Matlab) which is capable of solving non-linear models. It maybe that CAPE-OPEN is the key here, because CAPE-OPEN provides mechanism for integrating numerical method software into chemical engineering modelling software (Open Interface Specification Numerical Solvers, 1999).

The integration of more advanced methods with the OOP approach proposed will require more consideration.

8.8 An Alternative Modelling Goal

An alternative approach to the modelling goal could be to consider the dairy business from an environmental perspective (which could also be important in the profit context). As the influence of environmental factors on profit becomes significant (whether from artificial influences such as carbon credits, or more tangible factors like resource scarcity) the environmental focus may become an important part of the system (i.e. business) model. For example climatic factors are important to the long term look of the New Zealand raw milk supply curve, and they are also important contributors to the price of electricity and availability of water.

While energy and water consumption have always been important cost factors in the New Zealand dairy industry, until the mid 1990s they had been readily and cheaply available, and prices were relatively stable. As the industry (and the New Zealand economy) has grown, pressure has been placed on infrastructure and supply, so this is now changing. Prices that were once stable in the short term can now fluctuate significantly.

For example, where a long term fixed (i.e. steady state) power price model would have once sufficed, today only a dynamic model would be capable of providing quality data for even short term modelling. At the time of writing, New Zealand’s average annual power price was $25 - $30 per MWh (though the 3 month average to February 2006 was $90 per MWh). However, in the 2005 – 2006 summer spot prices reached as high as $270 per MWh (Gorman, 2006). Much of New Zealand’s electricity comes from hydro-generation. The consequence of low alpine rainfall is high power prices and stresses on both river and ground water supplies.

In New Zealand the environment can be a significant factor at a local level (primarily floods, drought, and earthquakes), and the climate is varied and occasionally harsh, the climate is reasonably predictable over the long term. Also the dairy industry has manufacturing flexibility, and for much of the year spare manufacturing capacity, so the effects of unusual climatic events which cause localized production shocks can be mitigated. In countries like Australia where extreme climatic conditions (caused by the El Niño Southern Oscillation) can be devastating to pastoral activities, it may be appropriate to include an environmental model as a partial model.

Monitoring and modelling of the environmental impact of industrial sites has in recent years taken on new significance, particularly regarding regulatory constraints and obligations, non-renewable resource consumption, and pollutant and effluent emissions. These three factors are important in the New Zealand context. Resource use consent must be obtained before water is extracted and effluent emitted – with legal limits being imposed on both.

The potential exists to extend, and improve the accuracy of, a profit focused system model by incorporating an environmental partial model into it.

9 Conclusion

Multiscale modelling as a practical tool for delivering decision making information is in its infancy. This work is the first known attempt to apply its concepts to dairy industry modelling. This is also the first known attempt to construct a multiscale production model from four partial models covering more than two scales.

By combining multiscale modelling theory, CAPE-OPEN specifications, and object oriented programming (OOP) concepts, a modelling and simulation tool has been developed using Microsoft’s Visual Basic .NET software development environment. This software tool is capable of being used to construct unit operation models, connect them into a sequential modular process model, and perform time incrementing simulations over the desired time frame, potentially extending out to years.

The following conclusions are drawn:

- OOP concepts used in conjunction with CAPE-OPEN specifications have a practical application in the implementation of multiscale models.
- multiscale modelling as applied here has a useful role in providing information to multiple decision making levels in the dairy industry.
- classifying partial model integration frameworks using the classification system proposed by Cameron et al. (2005) is not straightforward in all cases.
- the simulation’s incrementing regime, which controls the recalculation of partial models, has a significant effect on the performance of the system model. The number of unit operations in the process is also a factor in solution speed.
- a unit operation’s State property is a mechanism for integrating that model (at the macroscale) with microscale models.

OOP has distinct advantages in the multiscale modelling context. It readily facilitates the incremental addition of partial models to the system model. It also allows the modification of existing partial models without expensive modifications to other parts of the software. CAPE-OPEN provides, among other things, specifications and templates for the construction of unit operation models and their interconnection into process flow sheets. CAPE-OPEN also provides infrastructure definitions (such as ports, materials, port connection) and mechanisms (e.g. interface specifications) for the transfer of information between process scale, unit operation scale, and lower scale models. These are useful in the multiscale modelling context.

Using OOP to facilitate the construction of a sequential modular process model requires a different skill set from the model builder than the alternative equation oriented approach. Using the sequential modular modelling method the process model can be constructed intuitively by the process or production engineer who has knowledge of the process being modelled. Highly specialized modelling knowledge is not a requirement.

Other benefits of taking an OOP approach to the implementation of the multiscale model is demonstrated by the ease of construction of the model. Specifically, the ability to define unit operations with unique behavioral characteristics using generic software classes increases the flexibility of the model. OOP also facilitated rapid software development.

Though not done in this work, the basic structure has been developed in the modelling tool to allow compliance with the CAPE-OPEN specification to be implemented. CAPE-OPEN compliance would allow process modelling components and environments to be integrated with CAPE-OPEN compliant third-party software.

Using the cheese making process to demonstrate, the modelling tool has proved capable of delivering information to multiple decision making levels, including plant operators, plant managers, production managers, and sales and marketing managers. Though the model was a simplified representation of the cheese making process, information suitable for process troubleshooting, scheduling, optimization, and process control decision-making is generated.

The capacitive unit and flow unit object classes developed here were successfully able to be used to construct models for all the required unit operations in the cheese making process. These two classes are potentially powerful core objects which can be used as the basis for any unit operation model in any process. By implementing inheritance, it should be possible to construct more sophisticated unit operation models and construct industry specific modelling tools which are more intuitive to the chemical engineer.

It is evident that the cheese production model has the characteristics seen in multiscale models. Four partial models (i.e. unit operation material content, unit operation, raw milk supply, cheese making process) are integrated to create the production model. The partial models describing different levels of time, length and detail are integrated, and data can be generated for time scales ranging out to years.

In the time stepping mechanism used here to motivate model simulation, the iteration regime used to recalculate partial models significantly affected the performance of simulations. Because it is possible to iterate the simulation (and consequently recalculate the partial models) at fixed one second increments, solution times using this iteration regime are potentially too lengthy to be of practical use. This is especially true for long duration simulations (i.e. weeks and longer). This regime also resulted in extraneous data which is supplementary to reporting needs.

Increasing the fixed increment to speed up the simulation results in losses of accuracy. To overcome this problem, the discontinuity time step mechanism is a better method. The smallest time step which will result in the next important change in the process is found, and the next iteration of the simulation uses that time step. This method not only resulted in the fastest solution speed, but maintained the accuracy of the one second fixed increment method.

A simulation’s solution speed is also a function of the number of unit operations in the process. As more unit operations are added, computer processing demands increase and will slow the simulation.

The use of the unit operation’s State property is potentially a powerful mechanism to facilitate the construction of sophisticated unit operation models. The proposition is that the State could be used to define when a particular microscale model of a unit operation is to be recalculated. That is, a particular partial model would be recalculated when the unit operation is in a particular user-defined state. The example of this which has been implemented in this work is the recalculation of a capacitive unit operation’s material content when the unit operation is in the FILLING, EMPTYING, or FILLING/EMPTY state. Though a simple example, the potential of this mechanism is evident.

Further work on the model would be beneficial. Adding other microscale partial models to this model is desirable in order to further examine the usefulness of the State integration mechanism. This would also achieve better modelling accuracy, detail and data reporting capabilities. More work is also needed to refine unit operation behaviour, define interaction characteristics and examine connection possibilities which will provide more flexible and accurate process modelling. The construction of more sophisticated graphical user interfaces will make the model more user friendly while the implications of and methods for managing and distributing the large amounts of data which are generated needs analysis.

10 References

Akai, T. J., *Applied Numerical Methods for Engineers*, John Wiley and Sons, 1994.

Braunschweig, B. L., Pantelides, C. C., Britt, H. I., and Sama, S. (2000). Process Modelling; The promise of open software architectures, *Chemical Engineering Progress*, New York, **96 (**9), 65-77.

Bylund, G. (2003). *Tetra Pak Dairy Processing Handbook*, Tetra Pak Processing Systems AB.

Cameron, I. T., Ingram, G. D. and Hangos ,K.M. (2005). Chapter ??? – Multiscale process modelling. In *Computer Aided Process and Product Engineering* (Puigjaner, L. & Heyen, G. Eds), Wiley – VCH Verlag, Weinheim, Germany.

CAPE-OPEN Consortium (1999). *CAPE-OPEN Open Interface Specifications – Unit Operations* v2. Complete copies of CAPE-OPEN documentation are found at http://www.colan.org (last accessed 6 December 2005).

CAPE-OPEN Consortium (1999). *Open Interface Specification Numerical Solvers* 1.08. Complete copies of CAPE-OPEN documentation are found at http://www.colan.org (last accessed 6 December 2005).

CAPE-OPEN Consortium (2000). *Conceptual Design Document (CDD2)* v4. Complete copies of CAPE-OPEN documentation are found at http://www.colan.org (last accessed 6 December 2005).

Charpentier, J. C. (2003). Market Demand versus Technological Development: the Future of Chemical Engineering. *International Journal of Chemical Reactor Engineering* 1, Article A14.

Danish Dairy Board. http://www.mejeri.dk (last accessed 12 February 2006).

Drews, T. O., Krishnan, S., Alameda Jr, J. C., Gannon, D., Braatz, R. D. and Alkire, R. C. (2005). Multiscale simulations of copper electrodeposition onto a resistive substrate*. IBM Journal of Research and Development*, **49**, (1), 49 – 58.

Ennis, B. J, and Lister, J. D. *Perry's Chemical Engineer's Handbook* 7th ed., Perry, R. H., Green, D. W., Maloney, J. O. Editors, McGraw Hill, New York, 1997, 20-56 – 20-89.

Freeden, W., Michel, D. and Michel, V. (2004). Multiscale Modelling of Ocean Circulation. *Proc. Second International GOCE User Workshop "GOCE, The Geoid and Oceanography"*, ESA-ESRIN, Frascati, Italy, 8-10 March 2004 (ESA SP-569, June 2004).

Gorman, P. (21 January 2006). Breakdowns cause spot-price frenzy, *Christchurch Press*. pE4.

Ingram, G. D., Cameron, I. T. and Hangos, K. M. (2004). Classification and analysis of integrating frameworks in multiscale modelling. *Chemical Engineering Science* **59**, 2171 – 2187.

Jones, B. J. (1999). Cheese Process Control, *Master of Engineering Thesis,* Department of Chemical and Process Engineering, University of Canterbury.

Kurata, D. (2001). Object-Oriented Programming in Visual Basic, *Visual Studio .NET Technical Articles,* MSDN Library – Visual Studio .NET 2003.

Marquardt, W. (1995). Trends in Computer-Aided Process Modelling. *Computers and Chemical Engineering* **20** (6/7), 591 – 609.

Morison, K. R. (1997). Cheese Manufacture as a Separation and Reaction Process. *Journal of Food Engineering* **32**, 179 – 198.

Naysmith, M. and Douglas, P. L. (1995). Review of Real Time Optimization in Chemical Process Industries. *Developments in Chemical Engineering and Mineral Processing*, **3**, (2), 67 – 87.

Ng, K. M. (2004). MOPSD: A framework linking business decision-making to product and process design. *Computers and Chemical Engineering*, **29**, 51 – 56.

O’Callaghan, D. and O’Donnell, C. (1998). The Use of On-Line Sensors in Food Processing. Report DPRC No. 23, *Dairy Products Research Centre*, Moorepark, Fermoy, County Cork, Ireland.

Pattison, T. (2001). Using Inheritance in the .NET World. *Microsoft Developer Network (MSDN) Magazine.* November 2001 from Basic Instincts: Using Inheritance in the .NET World. MSDN Library – Visual Studio .NET 2003.

Preston, A. J. and Berzins, M. (1991). Algorithms for the Location of Discontinuities in Dynamic Simulation Problems, *Computers & Chemical Engineering,* **15**, 701 – 713.

Rey, A. D., Grecov, D. and Das, S. K. (2004). Thermodynamic and Flow Modelling of Meso- and Macrotexures in Polymer – Liquid Crystal Material systems. *Ind. Eng. Chem. Res.*, **43**, 7343 – 7355.

Srolovitz, D. J., Dandy, D. S., Butler, J. E., Battaile, C, C. and Paritosh (1997). The Integrated Multiscale Modelling of Diamond Chemical Vapour Deposition. *JOM* 49 (9) ABI / INFORM Trade and Industry, 42 – 47.

Ydstie, B. E. (2004). Distributed decision making in complex organizations: the adaptive enterprise. *Computers and Chemical Engineering*, **29**, 11 – 27.

11 Appendices

A Software Operating Instructions

The CD enclosed with this thesis contains the following files:

1. DecisionBridge.exe
   The software application developed to test the theories discussed in this thesis.

2. DecisionBridgeData.mdb
   The database to be used with DecisionBridge.exe. This database does not store simulation data.

<u>Disclaimer</u>

DecisionBridge was developed solely for the purposes of testing the ideas presented in this thesis. No responsibility is taken for any use of results generated by this software.

<u>System Requirements</u>

The DecisionBridge.exe application software has not undergone any systems or installation testing. However, it was developed on a Microsoft Windows XP system, with Microsoft Access 2003 installed and will function on that, or a compatible system. It is likely that the application will operate without Access installed (as earlier versions of Visual Basic did), but this has not been tested.

No minimum hardware requirements are specified. However, simulations were performed satisfactorily using a 1600MHz Intel Pentium M processor.

Installation Instructions

1. Create a folder called C:\Junk\
2. Copy both DecisionBridge.exe and DescisionBridgeData.mdb into C:\Junk\

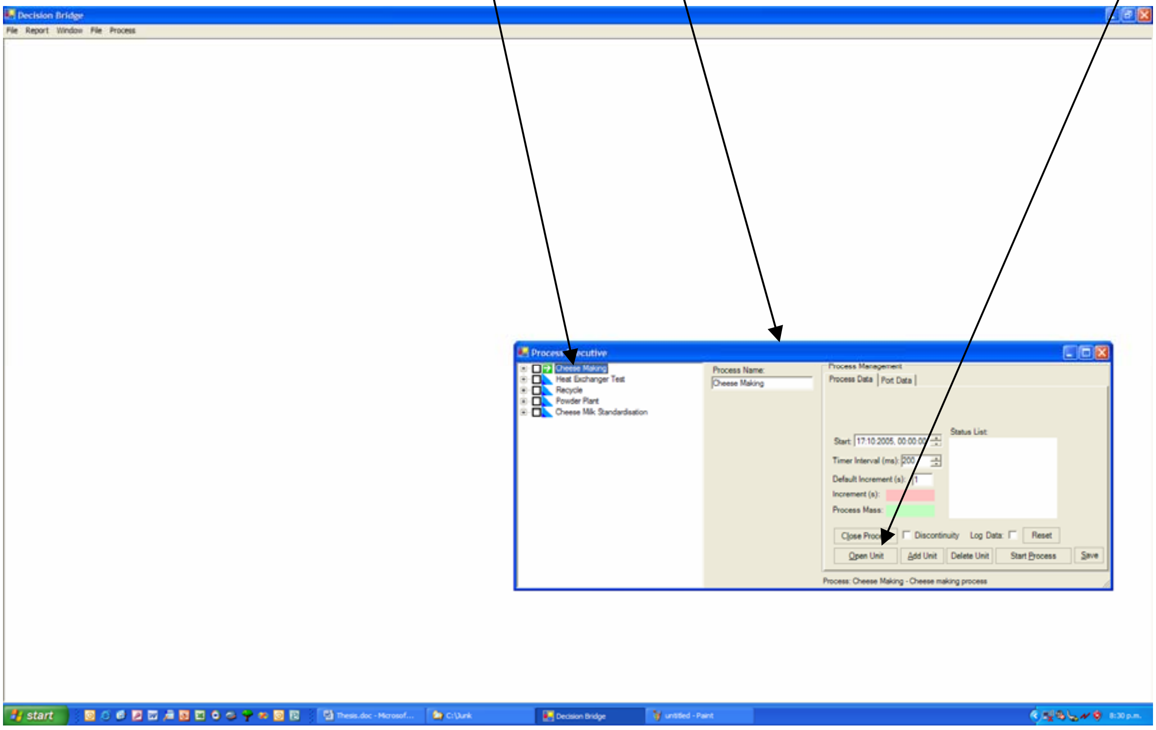
Operating Instructions

Follow these instructions to perform a simulation of a 2 day production run on the cheese making model.

1. Double click on DecisionBridge.exe. The MDI form will open with the Process Executive displayed.

   The cheese making process is selected on the left hand column of the process executive.

2. With the cheese making process selected, click on the “Open Unit” button. Click ‘Yes’ on the “Do you want to open all units?” message to open all the unit operation images.



Process Name: Cheese Making

Start: 17:10:2005, 00:00:00

Timer Interval (ms): 200

Default Increment (s): 1

Process: Cheese Making - Cheese making process

Figure A-1 – Opening Application Window Image

The application window should now look like Figure A-2.

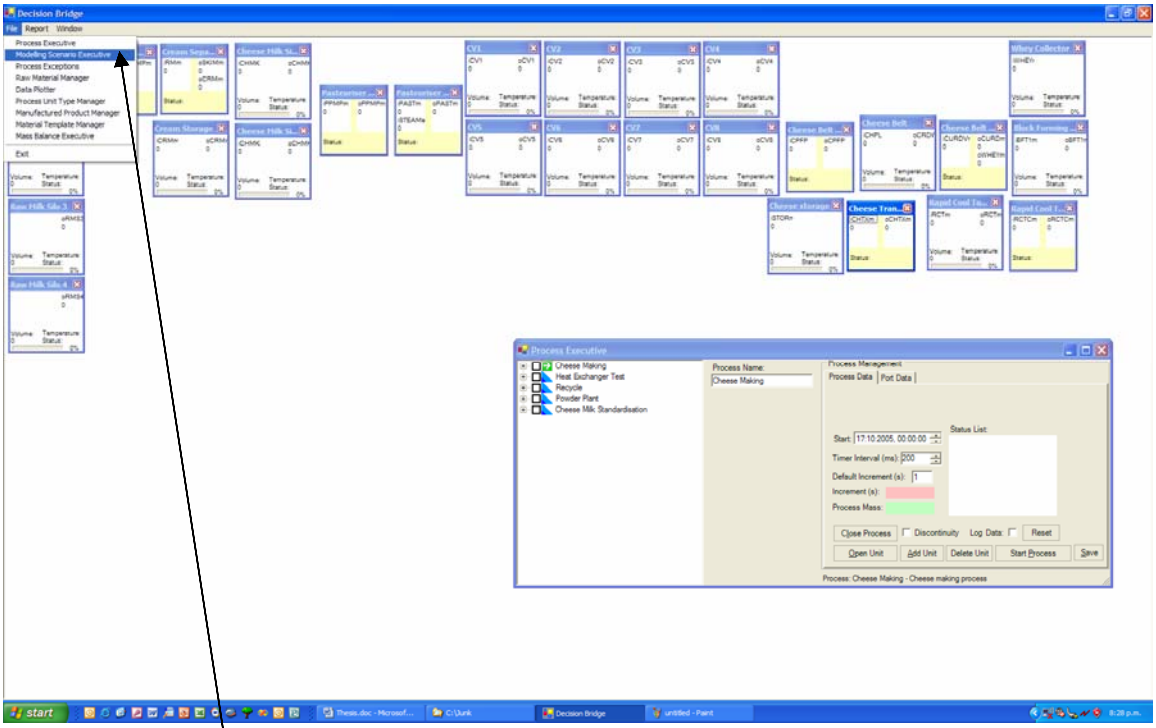


Figure A-2 –Application Window with Unit Operations

3. On the ‘File’ menu click on the “Modelling Scenario Executive” item to open the Modelling Scenario Executive form.

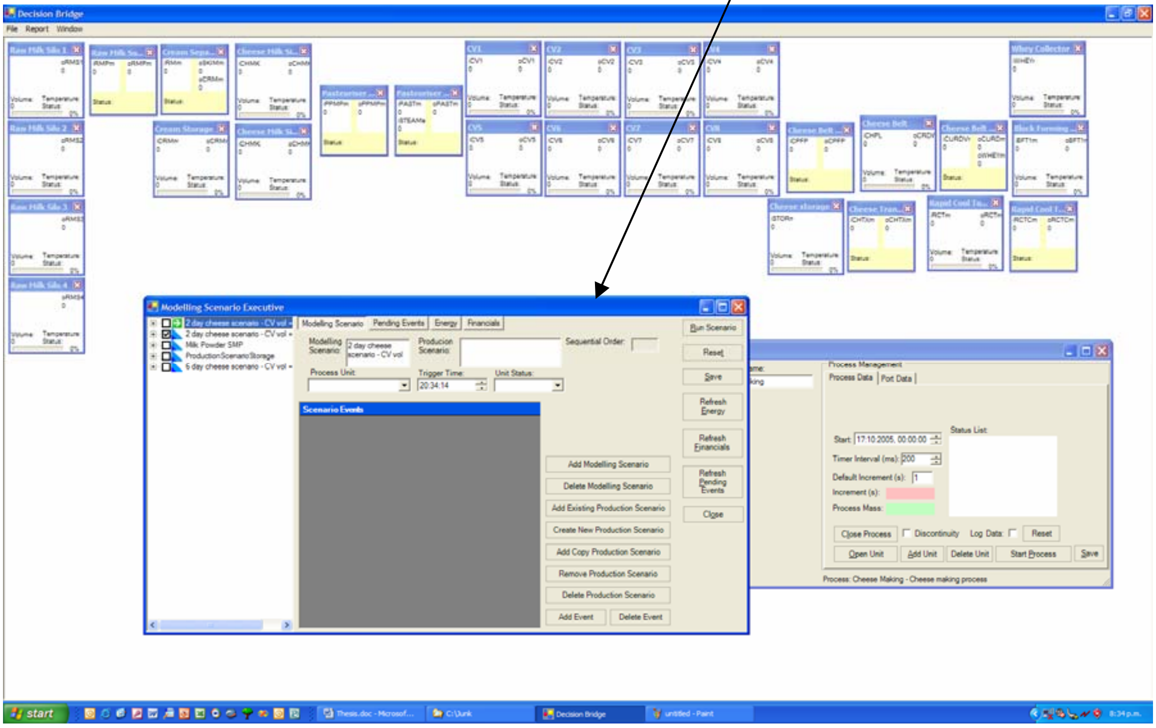
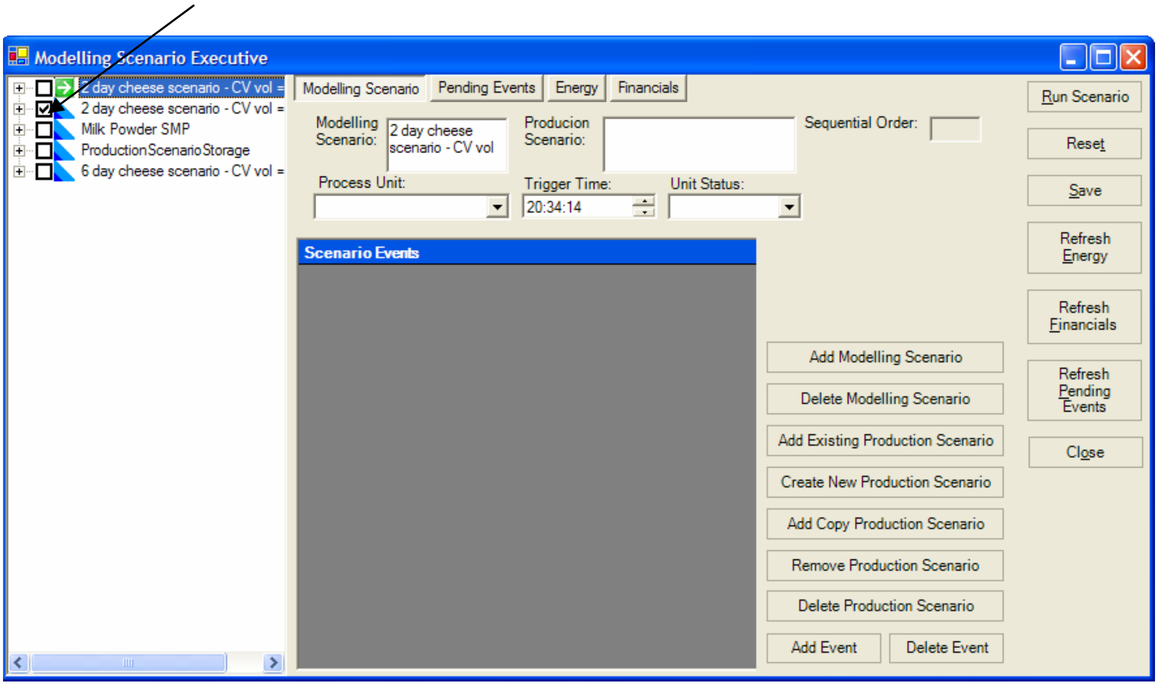


Figure A-3 –Application Window with Open Modelling Scenario Executive form

4. On the Modelling Scenario Executive form (Figure A-4) click on the second item to select the modelling scenario. A tick should appear in the check box next to the scenario name.


Modelling Scenario Executive
☐ 2 day cheese scenario - CV vol =
☑ 2 day cheese scenario - CV vol =
☐ Milk Powder SMP
☐ ProductionScenarioStorage
☐ 6 day cheese scenario - CV vol =
Modelling Scenario
Pending Events
Energy
Financials
Modelling Scenario:
Producion Scenario:
Sequential Order:
Process Unit:
Trigger Time:
Unit Status:
Scenario Events
Run Scenario
Reset
Save
Refresh Energy
Refresh Financials
Refresh Pending Events
Close
Add Modelling Scenario
Delete Modelling Scenario
Add Existing Production Scenario
Create New Production Scenario
Add Copy Production Scenario
Remove Production Scenario
Delete Production Scenario
Add Event
Delete Event

Modelling Scenario: 2 day cheese scenario - CV vol

Trigger Time: 20:34:14

Figure A-4 –Application Window with Open Unit Operations

5. Click the “Discontinuity” check box on the Process Executive to select it.

6. Click the “Run Scenario” button to start the simulation. The simulation will now run in discontinuity time increment mode. It will run until the 2 day simulation is complete (this should take a few minutes).

7. Once step 6 is complete, close the application and repeat the above steps, this time omitting Step 5. The simulation will now run in fixed time increment mode.

As the scenario runs, note the changing:

- port flowrates
- Unit operation state
- volume progress bar

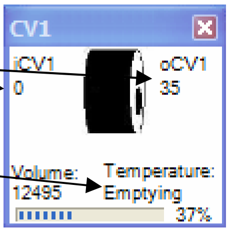
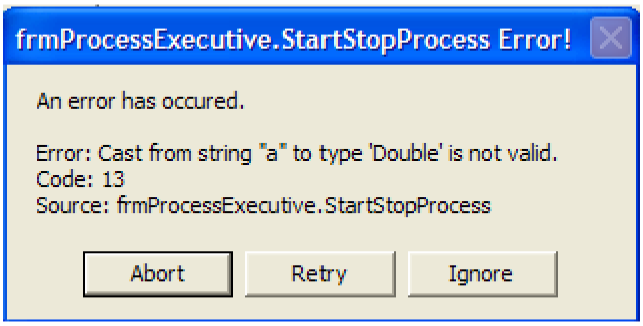


Figure A-5 –Capacitive Unit Operation

on the capacitive unit operation forms (Figure A-5).

If an message appears with the appearance of Figure A-6, click on the Abort button, close the application, and start from Step 1 again. This is the error handling mechanism in the software.



frmProcessExecutive.StartStopProcess Error!

An error has occured.

Error: Cast from string "a" to type 'Double' is not valid.
Code: 13
Source: frmProcessExecutive.StartStopProcess

Figure A-6 –Sample Error Message

B Unit Operation Volume Time Series Graphs

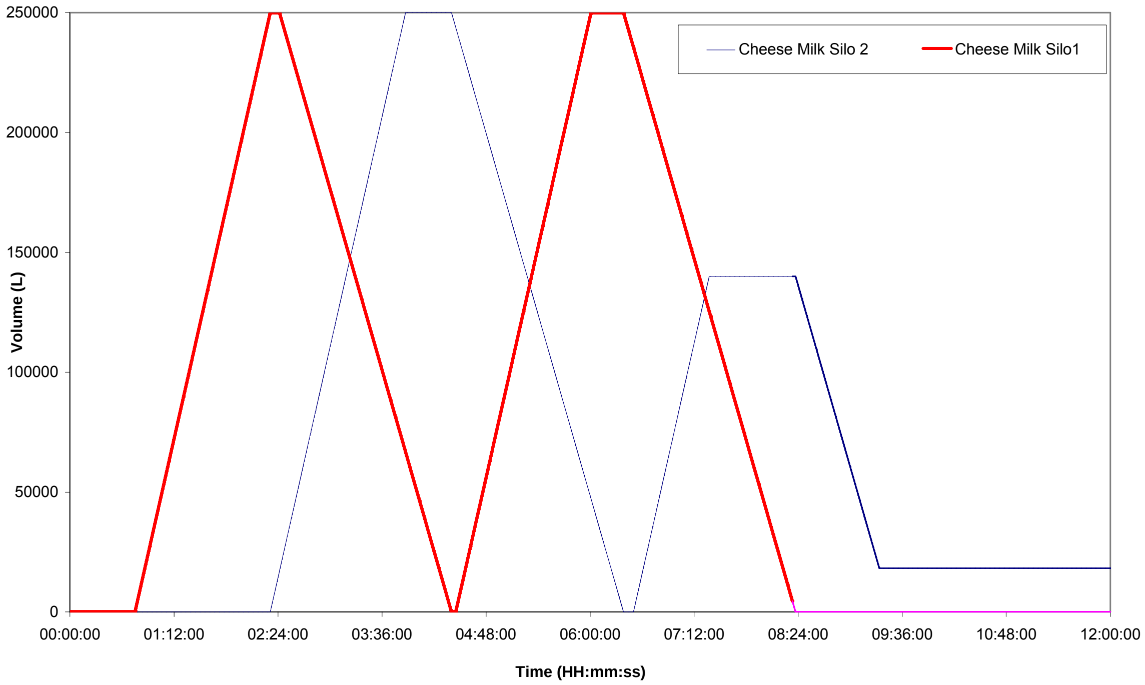


Figure B-1 – Cheese Milk Silos 1 & 2 – 12 hour volume time series – fixed time increment

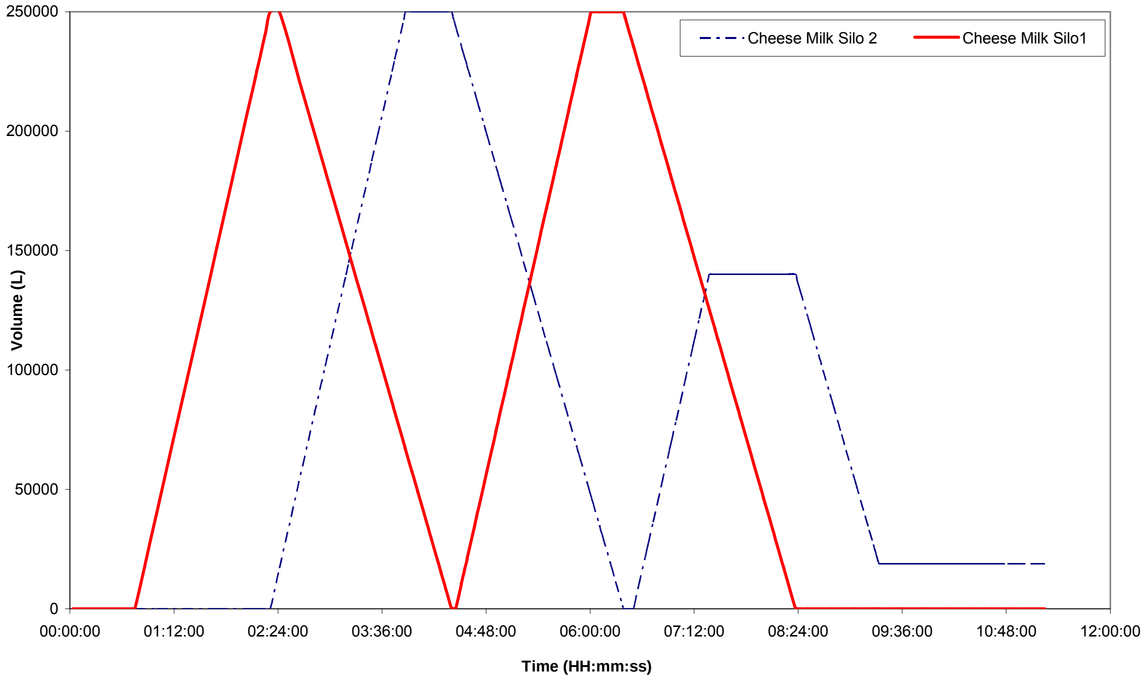


Figure B-2 – Cheese Milk Silos 1 & 2 – 12 hour volume time series – discontinuity time increment

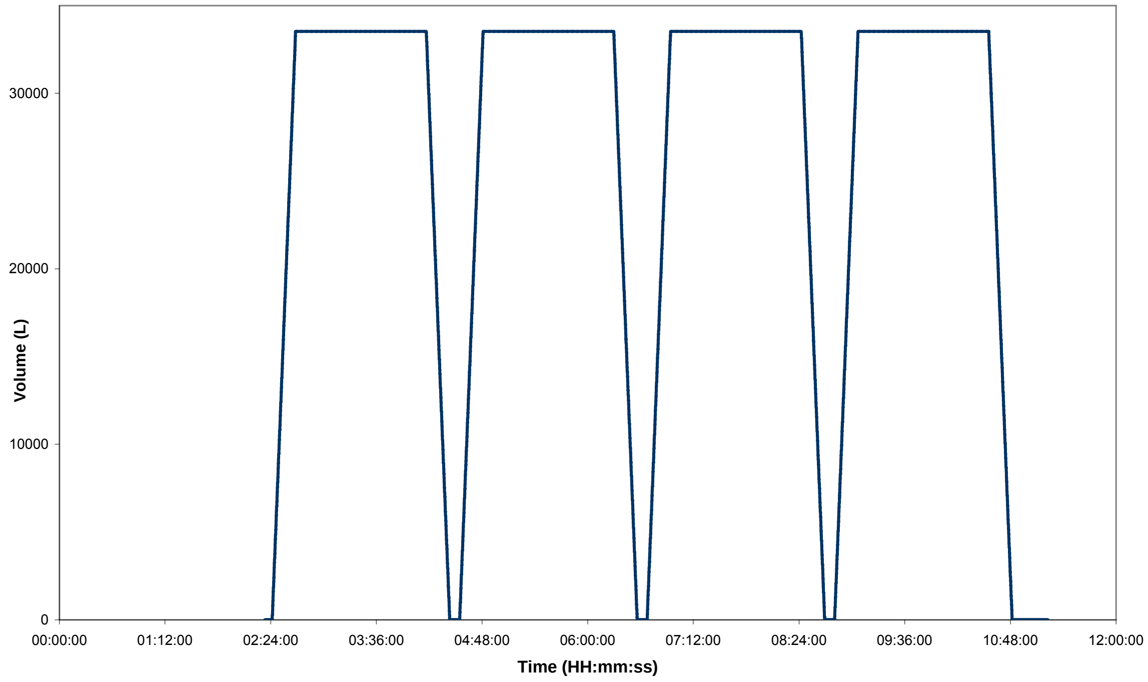


Figure B-3 – Cheese Vat 1 – 12 hour volume time series – fixed time increment

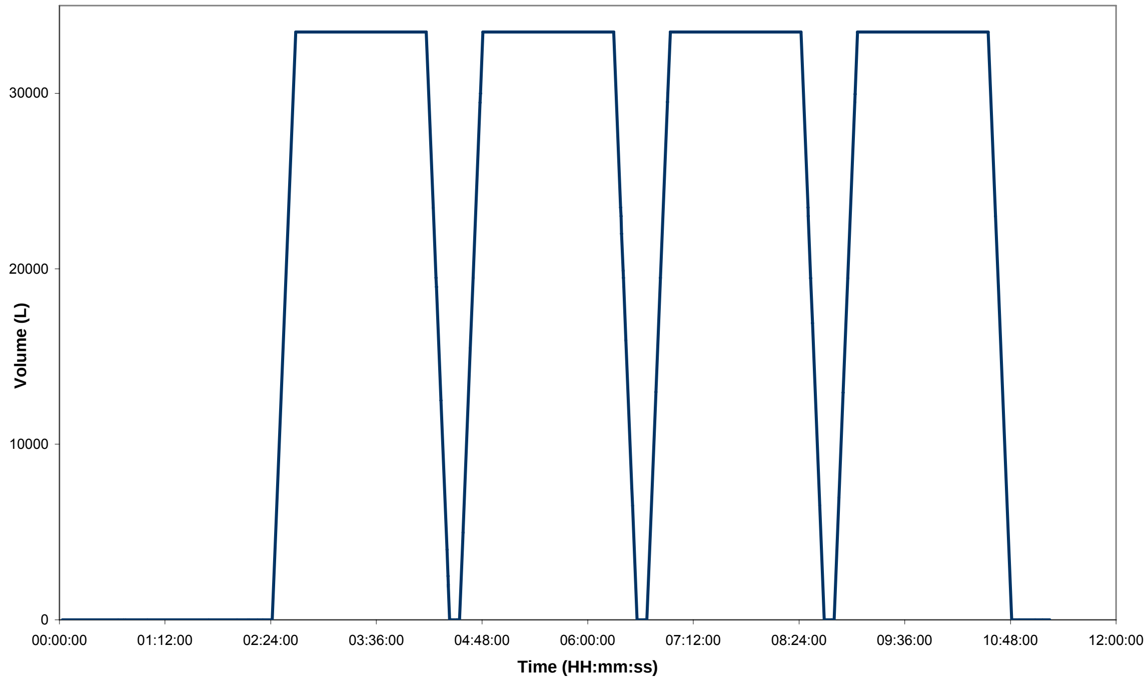


Figure B-4 – Cheese Vat 1 – 12 hour volume time series - discontinuity time increment

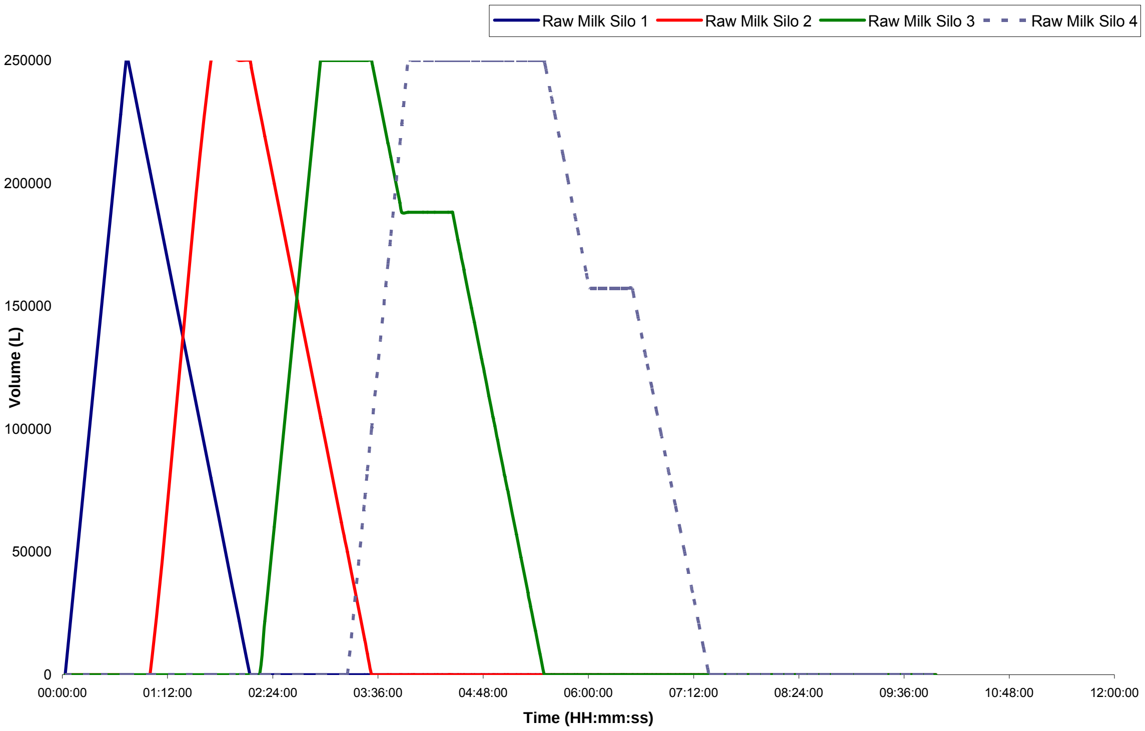


Figure B-5 – Raw Milk Silos 1, 2, 3 & 4 – 12 hour volume time series - discontinuity time increment

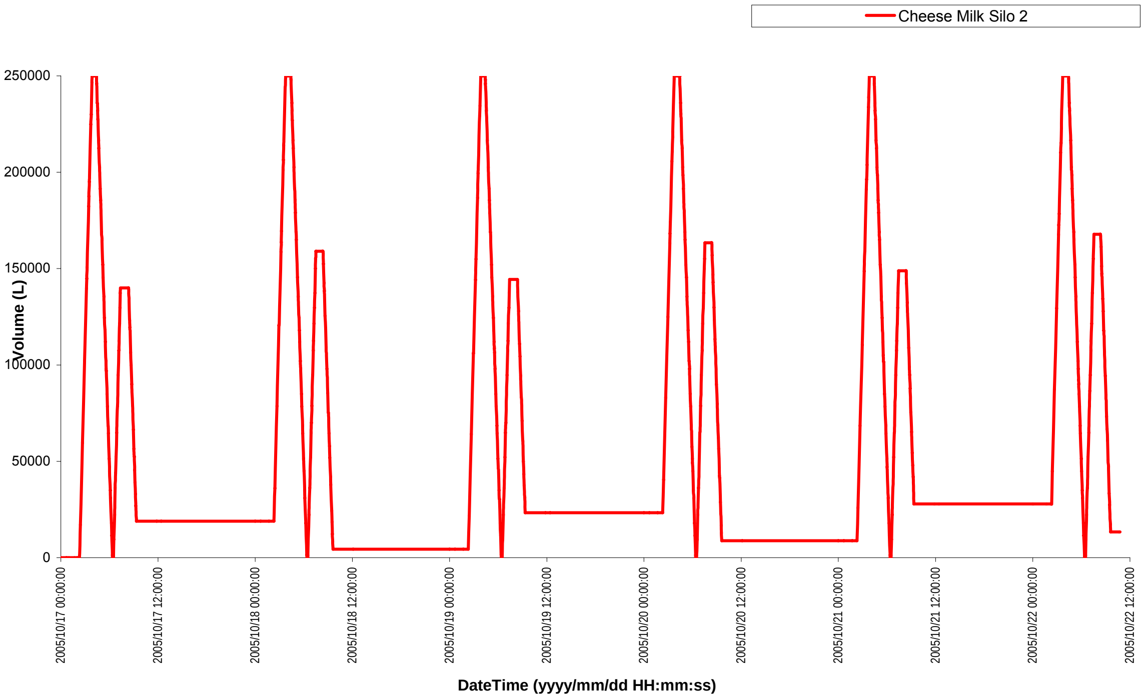


Figure B-6 – Cheese Milk Silos 1 – 6 day volume time series - discontinuity time increment

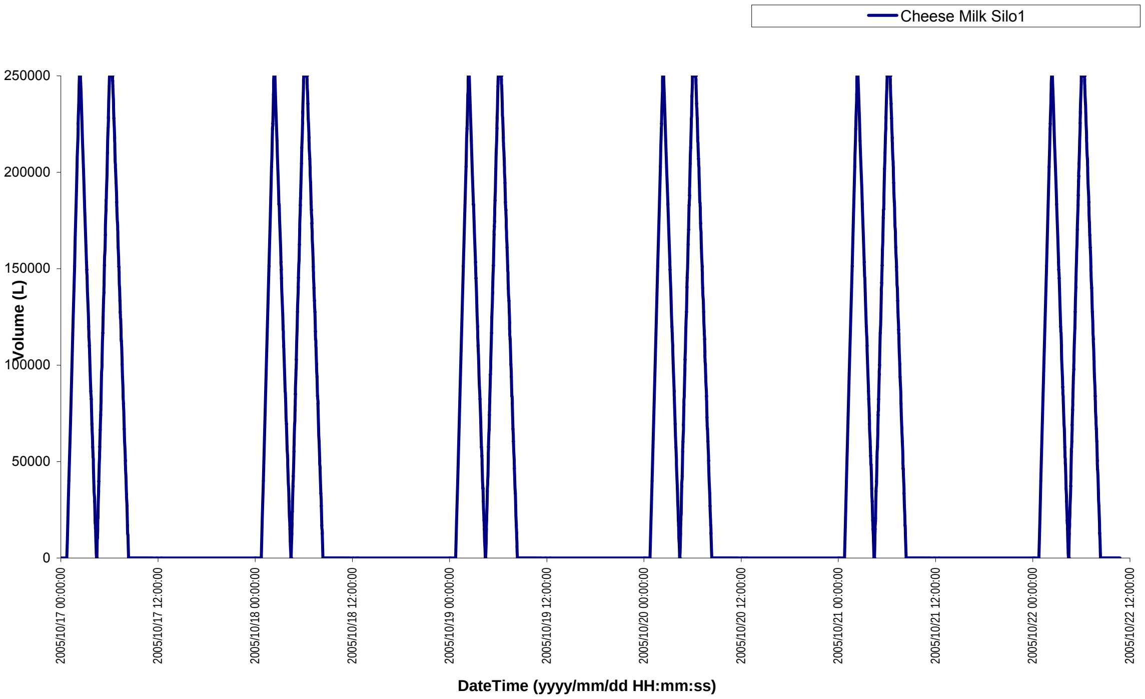


Figure B-7 – Cheese Milk Silos 1 – 6 day volume time series - discontinuity time increment

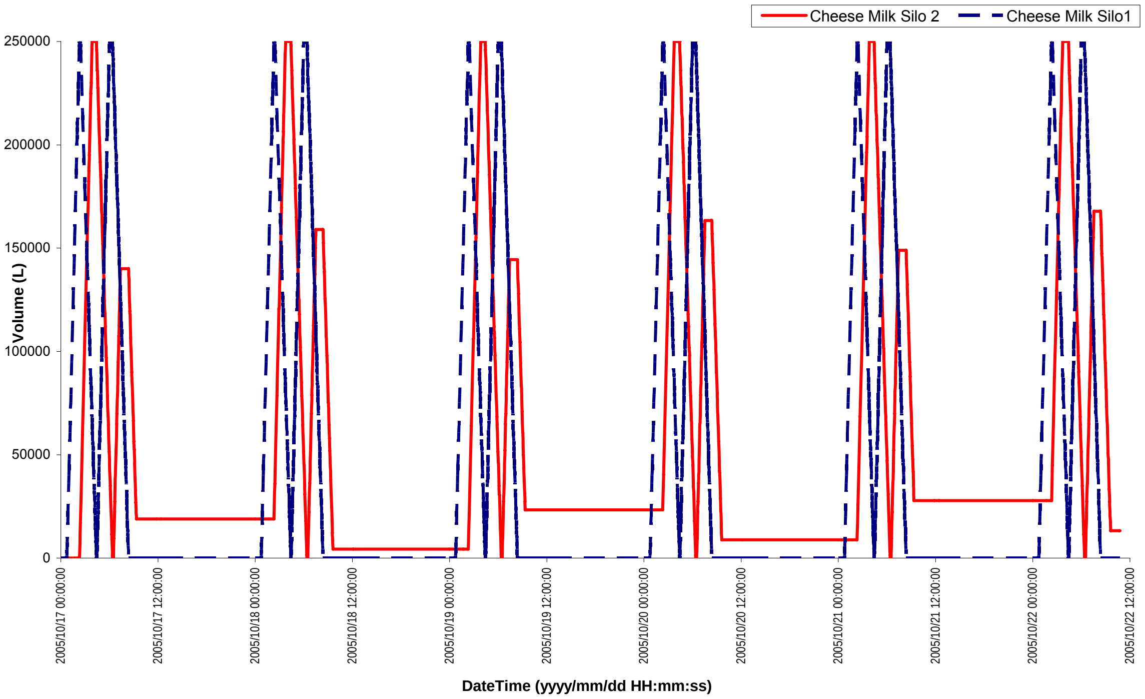


Figure B-8 – Cheese Milk Silos 1 & 2 – 6 day volume time series - discontinuity time increment

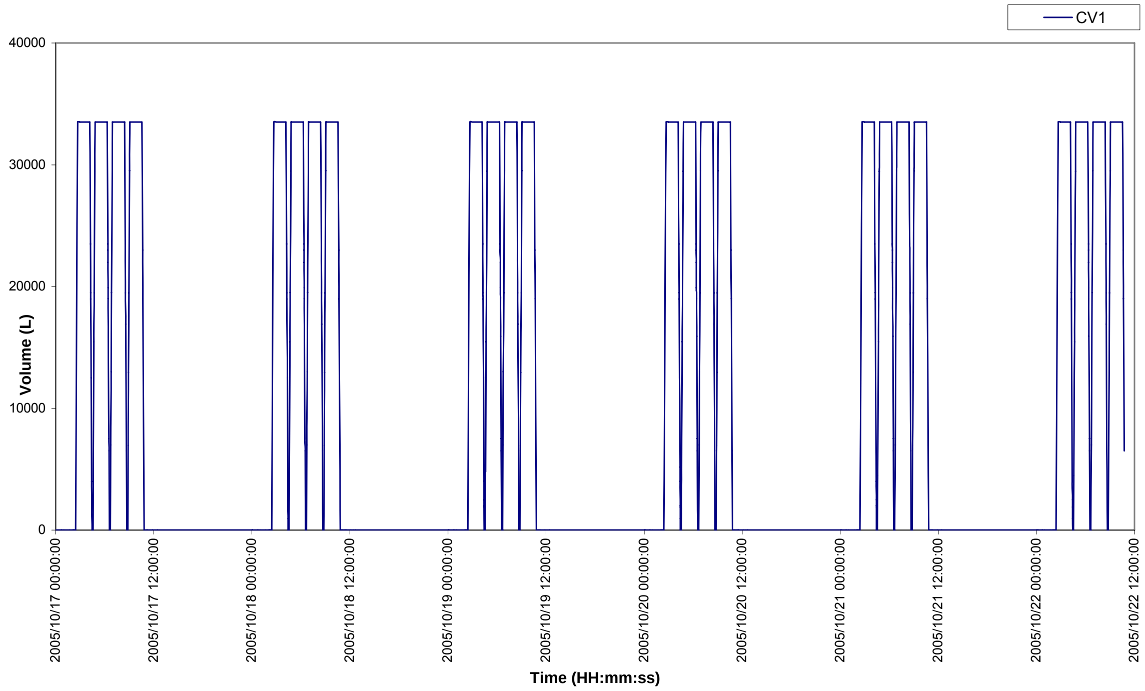


Figure B-9 – Cheese Vat 1 – 6 day volume time series - discontinuity time increment

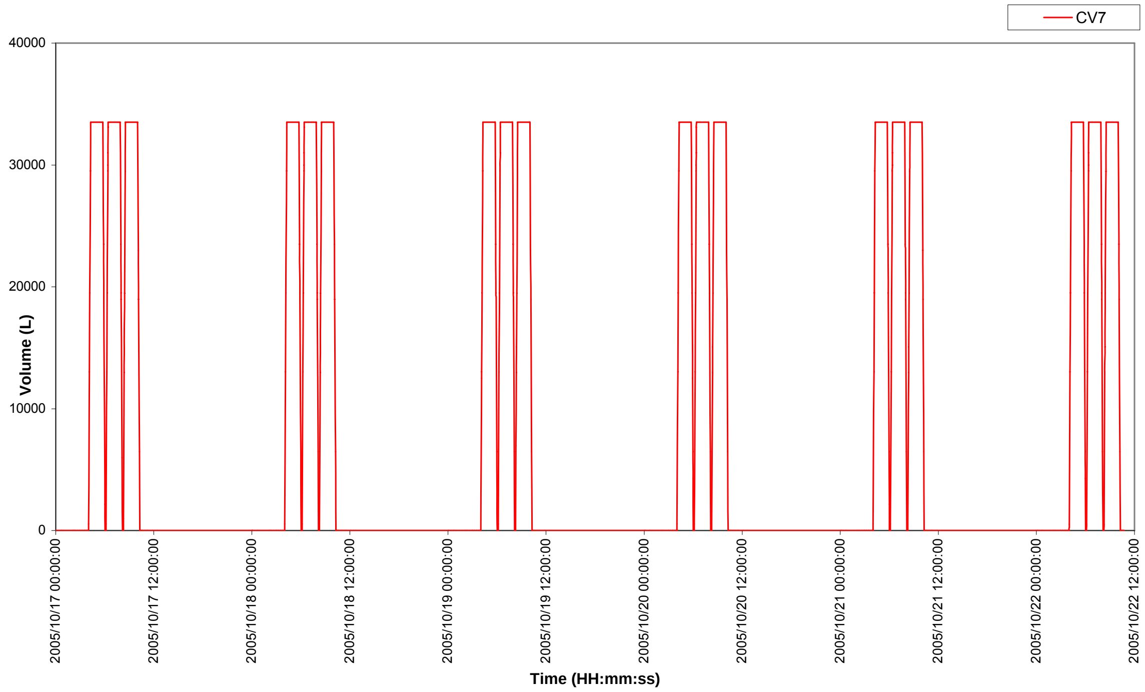


Figure B-10 – Cheese Vat 7 – 6 day volume time series - discontinuity time increment

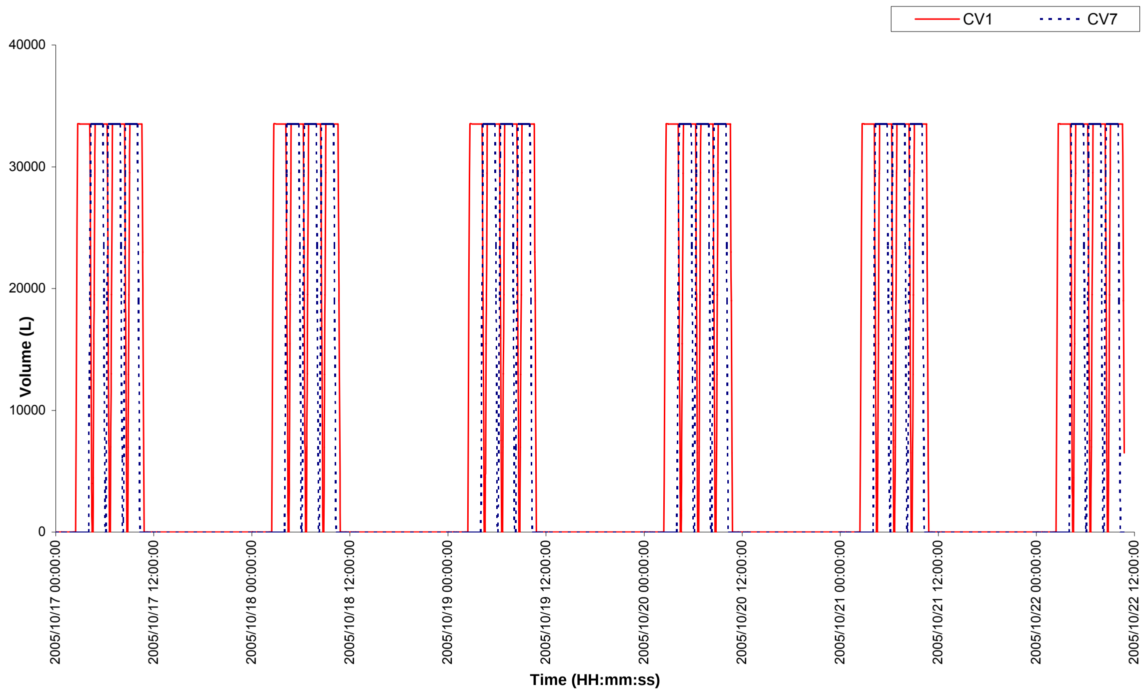


Figure B-11 – Cheese Vat 1 & 7 – 6 day volume time series - discontinuity time increment

C 6 Day Production Graph

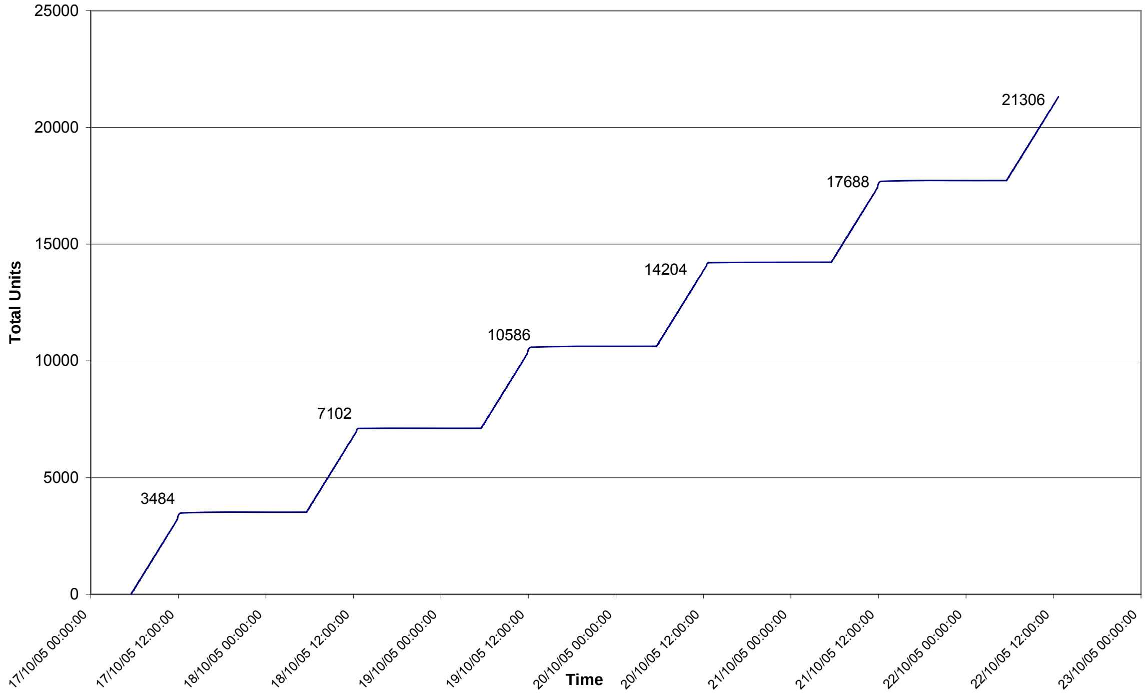


Figure C-1 – Total Manufactured Units of 25kg Bulk Cheddar – 6 day simulation

D Sample Object Code – Port Class

Software is developed in Visual Basic .NET.

This Port object is used below to demonstrate the general software coding structure of object classes used. Along with object properties (e.g. CollectionKey, PortKey, ProcessUnitKey, Tag), some of the Port's objects are shown (e.g. Material, Energy, Information, ConnectedPorts). The RecordStatus property is used to indicate if an object's properties have changed or if the object is new. When object data is being written to a database, only those objects whose record status is changed or new are saved.

```
Public Class Port
  Private zCollectionKey As String
  Private zRecordStatus As Integer
  Private zPortKey As Long
  Private zProcessUnitKey As Long
  Private zDirection As Long
  Private zPortType As Long
  Private zTag As String
  Private zDescription As String
  Private zFlowFraction As Single
  Private zFinalProductKey As Long
  Private zMaterial As Material
  Private zEnergy As Energy
  Private zInformation As Information
  Private zConnectedPorts As Ports

  Public Property RecordStatus() As Integer
    Get
      Return zRecordStatus
    End Get
    Set(ByVal Value As Integer)
      If zRecordStatus = RecordStatusEnum.gRecordAdd Then
        If Value = RecordStatusEnum.gRecordDelete Then zRecordStatus =
                RecordStatusEnum.gRecordAddDelete
          If Value = RecordStatusEnum.gRecordNoChange Then zRecordStatus = Value
        Else
          zRecordStatus = Value
        End If
      End Set
  End Property

  Public Property PortKey() As Long
    Get
      Return zPortKey
    End Get
    Set(ByVal Value As Long)
      zPortKey = Value
      RecordStatus = RecordStatusEnum.gRecordModify
    End Set
  End Property
  Public Property CollectionKey() As String
    Get
      Return zCollectionKey
    End Get
    Set(ByVal Value As String)
      zCollectionKey = Value
      RecordStatus = RecordStatusEnum.gRecordModify
    End Set
  End Property

Public Property ProcessUnitKey() As Long
    Get
      Return zProcessUnitKey
    End Get
    Set(ByVal Value As Long)
      zProcessUnitKey = Value
      RecordStatus = RecordStatusEnum.gRecordModify
    End Set
  End Property

  Public Property Direction() As Long
    Get
      Return zDirection
    End Get
    Set(ByVal Value As Long)
      zDirection = Value
      RecordStatus = RecordStatusEnum.gRecordModify
    End Set
  End Property

  Public Property PortType() As Long
    Get
      Return zPortType
    End Get
    Set(ByVal Value As Long)
    zPortType = Value
    RecordStatus = RecordStatusEnum.gRecordModify
    End Set
  End Property

  Public Property Description() As String
    Get
      Return zDescription
    End Get
    Set(ByVal Value As String)
      zDescription = Value
      RecordStatus = RecordStatusEnum.gRecordModify
    End Set
  End Property

  Public Property Tag() As String
    Get
      Return zTag
    End Get
    Set(ByVal Value As String)
      zTag = Value
      RecordStatus = RecordStatusEnum.gRecordModify
    End Set
  End Property

  Public Property FlowFraction() As Single
    Get
      Return zFlowFraction
    End Get
    Set(ByVal Value As Single)
      zFlowFraction = Value
      RecordStatus = RecordStatusEnum.gRecordModify
    End Set
  End Property

  Public Property FinalProductKey() As Long
    Get
      Return zFinalProductKey
    End Get
    Set(ByVal Value As Long)
      zFinalProductKey = Value
      RecordStatus = RecordStatusEnum.gRecordModify
    End Set
  End Property

  Public Property Material() As Material
    Get
      If zMaterial Is Nothing Then
        zMaterial = gMaterials.Item(PortKey.ToString)
      End If
      Return zMaterial
    End Get
    Set(ByVal Value As Material)
      zMaterial = Value
    End Set
  End Property

  Public Property Energy() As Energy
    Get
      If zEnergy Is Nothing Then
        zEnergy = gEnergys.Item(PortKey.ToString)
      End If
      Return zEnergy
    End Get
    Set(ByVal Value As Energy)
      zEnergy = Value
    End Set
  End Property

  Public Property Information() As Information
    Get
      If zInformation Is Nothing Then
        zInformation = gInformations.Item(PortKey.ToString)
      End If
      Return zInformation
    End Get
    Set(ByVal Value As Information)
      zInformation = Value
    End Set
  End Property

  Public ReadOnly Property PortConnections() As PortConnections
    Get
      Dim tmpPortConnections As PortConnections, tmpPortConnection As PortConnection
      Dim i As Integer
      Dim strCollKey As String

      tmpPortConnections = New PortConnections

      For Each tmpPortConnection In gPortConnections
        With tmpPortConnection
          strCollKey = .OutputKey & "_" & .InputKey
          If .InputKey = PortKey Or .OutputKey = PortKey Then
                    tmpPortConnections.Insert(tmpPortConnection, tmpPortConnection.CollectionKey)
          End If
        End With
      Next
      Return tmpPortConnections
    End Get
  End Property

  Public ReadOnly Property ConnectedPorts() As Ports
    Get
      Dim tmpPortConnection As PortConnection
      Dim tmpConnectedPort As Port

      If zConnectedPorts Is Nothing Then zConnectedPorts = New Ports
      zConnectedPorts.Clear()

      For Each tmpPortConnection In PortConnections
        tmpConnectedPort = Nothing
        If PortKey = tmpPortConnection.InputKey Then tmpConnectedPort =
                              Ports.Item(CStr(tmpPortConnection.OutputKey))
        Else
          tmpConnectedPort = gPorts.Item(CStr(tmpPortConnection.InputKey))
        End If
        If Not tmpConnectedPort Is Nothing Then
          zConnectedPorts.Insert(tmpConnectedPort, tmpConnectedPort.CollectionKey)
        End If
      Next
      Return zConnectedPorts
    End Get
  End Property

End Class
```

E Sample Collection Code – Ports Class

The Ports collection class is used to demonstrate the general structure of a collection.

```
Imports System.Collections
Public Class Ports
  Implements IEnumerable

  Private DeleteColl As Collection
  Private Coll As Collection


  Public Sub New()
    DeleteColl = New Collection
    Coll = New Collection
  End Sub

  Protected Overrides Sub Finalize()
    MyBase.Finalize()
    DeleteColl = Nothing
    Coll = Nothing
  End Sub

  Public ReadOnly Property Item(ByVal Index As Object) As Port
    Get
      Dim obj As Port
      On Error Resume Next
      obj = Coll.Item(Index)
      If Err.Number = 0 Then Return obj Else Return Nothing
    End Get
  End Property

  Public ReadOnly Property Count()
    Get
      Return Coll.Count
    End Get
  End Property

  Public ReadOnly Property DeleteCount()
    Get
      Return DeleteColl.Count
    End Get
  End Property

  Public Sub Clear()
    Coll = New Collection
    DeleteColl = New Collection
  End Sub


  Public Sub Remove(ByVal Index As Object)
    Coll.Remove(Index)
  End Sub

  Public Sub ClearDeleted()
    DeleteColl = New Collection
  End Sub

  Public Sub DeletePort(ByVal Index As Object)
    Dim tmpPort As Port
    tmpPort = Coll.Item(Index)
    If tmpPort.RecordStatus <> RecordStatusEnum.gRecordAdd Then
      DeleteColl.Add(tmpPort)
      Coll.Remove(Index)
    Else
      tmpPort.RecordStatus = RecordStatusEnum.gRecordAddDelete
    End If
    tmpPort = Nothing
  End Sub

  Public Function Add(Optional ByVal Key As Long = 0, Optional ByVal CollectionKey As String = "",
            Optional ByVal Before As Object = Nothing, Optional ByVal After As Object = Nothing) As Port
    Dim tmpPort As Port
    Dim lngKey As Long
    tmpPort = New Port
    If Key = 0 Then lngKey = NextKey() Else lngKey = Key
    If CollectionKey = "" Then CollectionKey = CStr(lngKey)
    Coll.Add(tmpPort, CollectionKey)
    With tmpPort
      .PortKey = lngKey
      .CollectionKey = CollectionKey
      .RecordStatus = RecordStatusEnum.gRecordAdd
    End With
    Return tmpPort
  End Function

  Public Sub Insert(ByVal inPort As Port, ByVal Key As String, Optional ByVal Before As Object = Nothing,
              Optional ByVal After As Object = Nothing)

    Coll.Add(inPort, CStr(Key), Before, After)
    inPort.CollectionKey = CStr(Key)
  End Sub

  Public Function GetEnumerator() As IEnumerator Implements IEnumerable.GetEnumerator
    Return Coll.GetEnumerator
  End Function

  Private Function NextKey() As Long
    Dim tmpConnection As New System.Data.OleDb.OleDbConnection
    Dim tmpAdapter As System.Data.OleDb.OleDbDataAdapter
    Dim dsKey As New System.Data.DataSet
    Dim dt As New DataTable
    Dim dr As DataRow
    Dim tmpPort As Port
    Dim intKey As Integer

    tmpConnection.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;" & _
              "Data Source=C:\Documents and Settings\Craig\My Documents\Masters
              Project\DecisionBridgeData.mdb;Mode=Share Deny None"

    tmpAdapter = New System.Data.OleDb.OleDbDataAdapter("Select * FROM NextKey WHERE
              TableName = 'Port'", tmpConnection.ConnectionString)

    dsKey = New System.Data.DataSet
    tmpAdapter.Fill(dsKey)
    dt = dsKey.Tables(0)
    dr = dt.Rows(0)
    intKey = dr.Item(1)
    dr.Item(1) = intKey + 1

    tmpAdapter.UpdateCommand = New OleDb.OleDbCommand("UPDATE NextKey SET NextKey = ?
              WHERE TableName = 'Port'", tmpConnection)
    tmpAdapter.UpdateCommand.Parameters.Add("@NextKey", OleDb.OleDbType.VarChar, 15,
              NextKey")
    tmpAdapter.Update(dsKey)
    dsKey.AcceptChanges()
    tmpConnection.Close()
    tmpConnection = Nothing
    Return intKey
  End Function

  Public Function GetPorts(Optional ByVal ProcessUnitKey As Long = 0) As Boolean
    Dim tmpConnection As New System.Data.OleDb.OleDbConnection
    Dim tmpAdapter As System.Data.OleDb.OleDbDataAdapter
    Dim dsPorts As New System.Data.DataSet
    Dim dt As New DataTable
    Dim dr As DataRow
    Dim tmpPort As Port
    Dim i As Int32, intKey As Int32
    Dim strSQL As String
    Dim tmpItemArray As Object
    GetPorts = False

    tmpConnection.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;" & _
              "Data Source=C:\Documents and Settings\Craig\My Documents\Masters
              Project\DecisionBridgeData.mdb;Mode=Share Deny None"

    strSQL = "Select * FROM Port"
    If ProcessUnitKey <> 0 Then
      strSQL = strSQL & " WHERE ProcessUnitKey = " & ProcessUnitKey.ToString
    End If

    tmpAdapter = New System.Data.OleDb.OleDbDataAdapter(strSQL, tmpConnection.ConnectionString)
    dsPorts = New System.Data.DataSet
    tmpAdapter.Fill(dsPorts)
    dt = dsPorts.Tables(0)

    For i = 0 To dt.Rows.Count - 1
      dr = dt.Rows(i)
      intKey = dr.ItemArray(0)
      tmpPort = Me.Add(intKey)
      With tmpPort
        .ProcessUnitKey = dr.ItemArray(1)
        .Direction = dr.ItemArray(2)
        .PortType = dr.ItemArray(3)
        .Tag = IIf(IsDBNull(dr.ItemArray(4)), "", dr.ItemArray(4))
        .FlowFraction = IIf(IsDBNull(dr.ItemArray(5)), 0, dr.ItemArray(5))
        .Description = IIf(IsDBNull(dr.ItemArray(6)), "", dr.ItemArray(6))
        .FinalProductKey = IIf(IsDBNull(dr.ItemArray(7)), 0, dr.ItemArray(7))
        .RecordStatus = RecordStatusEnum.gRecordNoChange
      End With
    Next

    tmpConnection.Close()
    tmpConnection = Nothing
    Return True
  End Function

  Public Function Update(Optional ByVal Key As Integer = Nothing) As Boolean
    Dim tmpConnection As New System.Data.OleDb.OleDbConnection
    Dim tmpAdapter As System.Data.OleDb.OleDbDataAdapter
    Dim ds As New System.Data.DataSet
    Dim dt As New DataTable, dr As DataRow
    Dim tmpPort As Port
    Dim intKey As Integer
    Dim strSQL As String
    Update = False

    tmpConnection.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;" & _
              "Data Source=C:\Documents and Settings\Craig\My Documents\Masters
              Project\DecisionBridgeData.mdb;Mode=Share Deny None"
```

```
If Not DeleteColl.Count = 0 Then
  For Each tmpPort In DeleteColl
    tmpAdapter = New System.Data.OleDb.OleDbDataAdapter("Select * FROM Port WHERE PortKey
          = " & tmpPort.PortKey, tmpConnection.ConnectionString)
    With tmpAdapter
      .Fill(ds)
      dt = ds.Tables(0)
      dr = dt.Rows(0)
      dr.Delete()
      strSQL = "DELETE FROM Port WHERE PortKey = " & tmpPort.PortKey
      .DeleteCommand = New OleDb.OleDbCommand(strSQL, tmpConnection)
      .DeleteCommand.Parameters.Add("@PortKey", OleDb.OleDbType.VarChar, 15, "PortKey")
      .Update(ds)
      tmpAdapter = Nothing
      ds.AcceptChanges()
      ds.Clear()
    End With
  Next
  ClearDeleted()
End If

For Each tmpPort In Coll
  If tmpPort.RecordStatus = RecordStatusEnum.gRecordModify Or tmpPort.RecordStatus =
                RecordStatusEnum.gRecordAdd Then
tmpAdapter = New System.Data.OleDb.OleDbDataAdapter("Select * FROM Port WHERE PortKey
                = " & tmpPort.PortKey, tmpConnection.ConnectionString)
    With tmpAdapter
      .Fill(ds)
      dt = ds.Tables(0)
      Select Case tmpPort.RecordStatus
        Case RecordStatusEnum.gRecordNoChange
        Case RecordStatusEnum.gRecordAddDelete
        Case RecordStatusEnum.gRecordDelete
        Case RecordStatusEnum.gRecordModify

        dr = dt.Rows(0)
        If dr.Item(1) <> tmpPort.ProcessUnitKey Then dr.Item(1) = tmpPort.ProcessUnitKey
        If dr.Item(2) <> tmpPort.Direction Then dr.Item(2) = tmpPort.Direction
        If dr.Item(3) <> tmpPort.PortType Then dr.Item(3) = tmpPort.PortType
        If IIf(IsDBNull(dr.Item(4)), "", dr.Item(4)) <> tmpPort.Tag Then dr.Item(4) = tmpPort.Tag
        If dr.Item(5) <> tmpPort.FlowFraction Then dr.Item(5) = tmpPort.FlowFraction
        If IIf(IsDBNull(dr.Item(6)), "", dr.Item(6)) <> tmpPort.Description Then dr.Item(6) =
                tmpPort.Description
        If IIf(IsDBNull(dr.Item(7)), 0, dr.Item(7)) <> tmpPort.FinalProductKey Then dr.Item(7) =
                tmpPort.FinalProductKey

        strSQL = "UPDATE Port SET ProcessUnitKey = ?, Direction = ?, PortType = ?, Tag = ?,
                FlowFraction = ?, Description = ?, FinalProductKey = ? WHERE PortKey = " &
                tmpPort.PortKey

        .UpdateCommand = New OleDb.OleDbCommand(strSQL, tmpConnection)
        With .UpdateCommand.Parameters
          .Add("@ProcessUnitKey", OleDb.OleDbType.VarChar, 15, "ProcessUnitKey")
          .Add("@Direction", OleDb.OleDbType.VarChar, 15, "Direction")
          .Add("@PortType", OleDb.OleDbType.VarChar, 15, "PortType")
          .Add("@Tag", OleDb.OleDbType.VarChar, 10, "Tag")
          .Add("@FlowFraction", OleDb.OleDbType.VarChar, 15, "FlowFraction")
          .Add("@Description", OleDb.OleDbType.VarChar, 255, "Description")
          .Add("@FinalProductKey", OleDb.OleDbType.VarChar, 15, "FinalProductKey")
        End With
        tmpPort.RecordStatus = RecordStatusEnum.gRecordNoChange

      Case RecordStatusEnum.gRecordAdd
        dr = dt.NewRow
        dr(0) = tmpPort.PortKey
        dr(1) = tmpPort.ProcessUnitKey
```

```
            dr(2) = tmpPort.Direction
            dr(3) = tmpPort.PortType
            dr(4) = tmpPort.Tag
            dr(5) = tmpPort.FlowFraction
            dr(6) = tmpPort.Description
            dr(7) = tmpPort.FinalProductKey
            dt.Rows.Add(dr)

            strSQL = "INSERT INTO Port(PortKey, ProcessUnitKey, Direction, PortType, Tag,
                    FlowFraction, Description, FinalProductKey) VALUES (?, ?, ?, ?, ?, ?, ?, ?)"

            .InsertCommand = New OleDb.OleDbCommand(strSQL, tmpConnection)

            With .InsertCommand.Parameters
              .Add("@PortKey", OleDb.OleDbType.VarChar, 15, "PortKey")
              .Add("@ProcessUnitKey", OleDb.OleDbType.VarChar, 15, "ProcessUnitKey")
              .Add("@Direction", OleDb.OleDbType.VarChar, 15, "Direction")
              .Add("@PortType", OleDb.OleDbType.VarChar, 15, "PortType")
              .Add("@Tag", OleDb.OleDbType.VarChar, 6, "Tag")
              .Add("@FlowFraction", OleDb.OleDbType.VarChar, 15, "FlowFraction")
              .Add("@Description", OleDb.OleDbType.VarChar, 255, "Description")
              .Add("@FinalProductKey", OleDb.OleDbType.VarChar, 15, "FinalProductKey")
            End With
          End Select

          .Update(ds)
          tmpAdapter = Nothing
          ds.AcceptChanges()
          ds.Clear()
          tmpPort.RecordStatus = RecordStatusEnum.gRecordNoChange
        End With
      End If
    Next

    tmpConnection.Close()
    tmpConnection = Nothing
    Return True
  End Function
End Class
```