A SERVICE-ORIENTED APPROACH FOR DEVELOPING ADAPTIVE DISTRIBUTION CHAIN

Reggie Davidrajuh

Department of Electrical & Computer Engineering,

University of Stavanger,

PO Box 8002, 4036 Stavanger, Norway

Email: reggie.davidrajuh@uis.no

Abstract: This paper presents a service-oriented approach for development of adaptive distribution chain. The distribution chain presented in this paper achieves all the three objectives of an adaptive chain: Firstly, to achieve variability, this paper proposes an approach that uses iterations to fine-tune the modules that makeup the distribution chain. Secondly, to achieve visibility, this paper proposes Web services based Service Component Architecture for implementation of the distribution chain. Finally, mathematical modelling and simulation of the distribution chain is proposed for assuring the velocity.

Keywords: Adaptive distribution chain, service-oriented approach, performance analysis, Petri Net, GPenSIM

Bibliographical notes: Reggie Davidrajuh received a PhD in Industrial Engineering in 2000 and a Masters Degree in Control Systems in 1994, both from the Norwegian University of Science and Technology. He is currently an Associate Professor of Computer Science at the University of Stavanger, Norway.

1. INTRODUCTION

This paper presents a service-oriented approach for development of adaptive distribution chain. First, this paper talks about the design issues of an adaptive distribution chain, then about the implementation issues. Finally, performance analysis of an adaptive distribution chain is discussed.

Literature presents many definitions for adaptive supply (or distribution) chain; a formal definition of an adaptive supply (or distribution) chain is: “An adaptive supply chain is one in which participants are confident in their ability to recognize changing or unanticipated conditions in their supply chain, in a timeframe that allows them to evaluate alternative corrective actions, and react to mitigate the impact to their business. Due to the frequency and volume of supply chain exceptions, the ability to monitor and respond must be highly efficient and at least partly automated. The amount of reserve inventory in a supply chain is usually indirectly proportional to the confidence operators have in their ability to recover from supply chain failures without significant impact to schedules, delivery dates, or costs” (World Trade, 2004).

The objective of an adaptive supply chain is to have the highest *visibility*, greatest *velocity*, and best ability to manage *variability* (i2, 2004; SAP, 2002). Visibility is the ability of all the collaborating partners to see vital data; velocity is the speed of information flow across the supply chain; variability is the ability to withstand unpredictable events. Achieving these three objectives is not easy, considering the complexity and diversity of the applications collaborating partners are using (Smith, 2006; Shih et al, 2006; Wei at al, 2006). Thus, this paper presents a unique design approach for distribution chain that is based service component architecture, consisting modular components that are attuned by iterations.

Organisation of this paper:

The distribution chain presented in this paper achieves all the three objectives (variability, visibility, and velocity) of an adaptive chain. Section 2 presents a unique design approach for achieving variability. To achieve visibility, this paper proposes Web services based Service Component Architecture (SCA) for implementation of the distribution chain; this is explained in section 3. How fast the information flows across the distribution chain (velocity) is calculated by means of modelling and simulation. Section 4 presents the details of the performance analysis.

2. MODULAR COMPONENTS BASED DESIGN

This section presents a modular component based design to achieve variability - the ability of the distribution chain to be adaptive to the quickly changing market conditions. The design mainly involves an iterative cycle connecting two modules: the strategic module and the tactical module; see figure 1. The tactical module is further divided into four distinct modules: two modules for inventory control and two modules for transport. The author's earlier paper Ma & Davidrajuh (2005) presents complete mathematical details of all these modules; a summary is given in the following subsections.

2.1 The Iterations

In the initialization module, all parameters (including operation related parameters) are assigned initial values from the ongoing (and/or from the past) collaboration. Then, in the strategic module, some strategic decisions are made (e.g. optimal number and location of wholesalers and retailers). After this, from the output of strategic module, the tactical decisions are made (e.g. optimal transportation schedules & routes, inventory control parameters) in the tactical module.

After determining the tactical parameters, the parameter values are input into the strategic module again, starting another re-optimization cycle. Successive design results are compared at the end of each iteration-cycle. If there is no remarkable difference found between the successive iteration results, this means that the iteration results converge and the distribution chain is established.

When the market conditions change, there may be changes in the values of some of the parameters such as demand, transportation costs, etc. First, for these new values, the tactical module is invoked (operation related parameters are re-calculated) to adapt to the new market situation. If the resulting operation related parameter values significantly differ from the older values, then the nucleus enterprise may start the iteration cycles until the iteration result converges: at this point, a new distribution chain is found that is adapted to the new market.

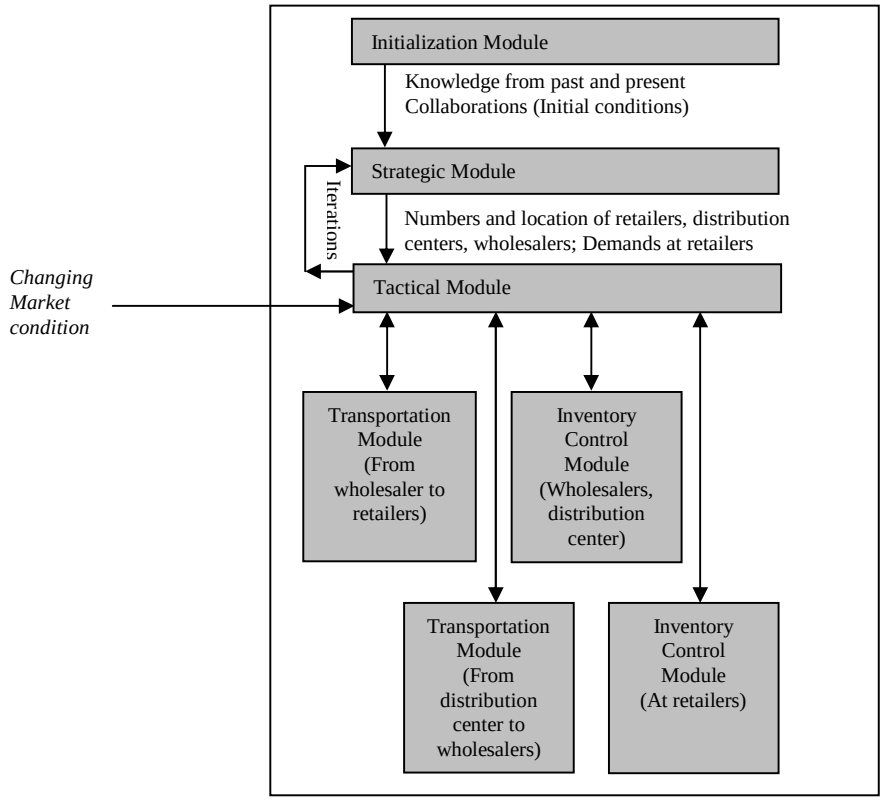

Figure 1: Achieving variability: the iterative approach with modular components

2.2 The Strategic Module

The strategic module is about finding optimal number and location of the collaborators, as the performance of a distribution chain is mainly dependent on its structure (Caputo et al, 2004; Childerhouse et al, 2003) and on the relationship that exist between the collaborators (Rahman, 2004; Wu et al, 2004).

The strategic module is modelled with mixed integer programming (MIP) composed of two types of formulae: objective function and constraints. The objective function of the strategic module is to maximize profit, while the constraints are like flexibility constraint, material flow balance constraint, etc. In the objective function, *profit* equals total revenue minus total cost; only four types of costs are considered: delivery cost from wholesalers to retailers, delivery cost from distribution centres to wholesalers, inventory holding cost at wholesalers, and inventory cost at the distribution centre.

2.3 The Tactical Module

The tactical module is determined based on the output of the strategic module. The tactical module has the following four sub-modules:

The sub-module for inventory control at retailers: According to Tijms (1994), there are mainly two types of inventory control models: periodic review model and continuous review model. Ma and Davidrajuh (2005) propose use of continuous review model for the inventory control policy.

Sub-module for transportation from a wholesaler to retailers: After determining optimal inventory levels for each retailer, transportation between wholesalers and retailers can be planned. In practice, the order quantity by a retailer is normally small, so it is possible for a vehicle to serve several retailers in one journey. In such situation, following questions are raised: How to cluster retailers? How to determine routes for vehicles? Ma and Davidrajuh (2005) propose use of genetic algorithm for routing.

Sub-module for inventory control at wholesalers and at distribution centre: A wholesaler faces several retailers; adding together random demands at retailers can generate the random demand process at the wholesaler. For the distribution centre, the parameter determining process is same as the one for wholesalers.

Sub-module for transportation from distribution centre to wholesalers: Normally, amount of product demanded by a wholesaler is large and hence, a vehicle can only serve one wholesaler in its journey. Thus, there is no routing problem in this transportation model (if there is routing problem, then the mathematical model used for the sub-module "Transportation model from a wholesaler to its retailers" can be used).

3. SERVICES-ORIENTED IMPLEMENTATION

The design presented in section 2 (summarized in figure 1) involves modules for achieving variability. The same components can be utilized to achieve visibility too; this paper proposes

Service Component Architecture (SCA) for implementation of the modules. Figure 2 shows how visibility can be added to the distribution chain by implementing modules as services. SCA is based on Web services. Beatty et al (2003) and Beisiegel et al (2005) explain the concepts behind SCA.

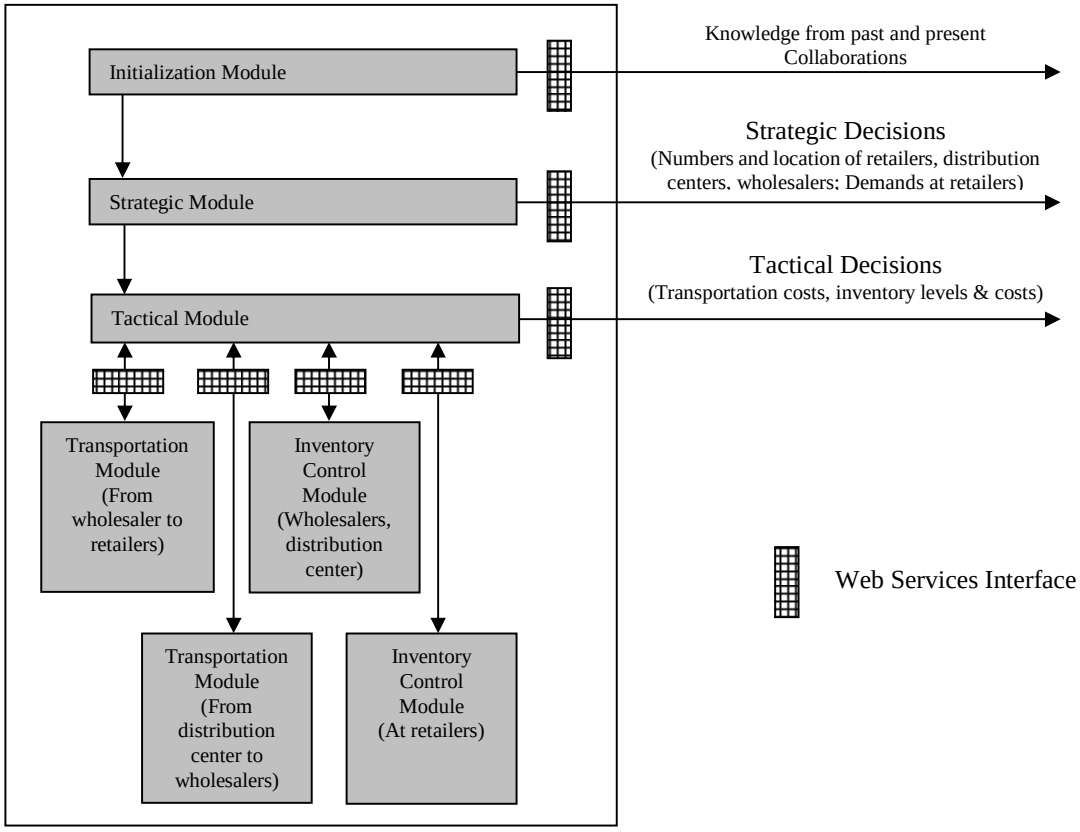

Figure 2: Achieving visibility with Web services

3.1 A Brief Introduction to Service Component Architecture (SCA)

A model of a business application (known as 'SCA Assembly Model') consists of a series of artefacts: The basic artefact is the '*module*', which is the unit of deployment for SCA and which holds '*services*' that can be accessed remotely. A module contains one or more '*components*', which contain the business logic (function) provided by the module. Components offer their function as services, which can either be used by other components within the same module or which can be made available for use outside the module through '*entry points*'. Components may also depend on services provided by other components - these dependencies are called '*references*'. References can either be linked to services provided by other components in the same module, or references can be linked to services provided outside the module, which can be provided by other modules. References to services provided outside the module, including services provided by other modules, are defined by '*External Services*' in the module. '*Wires*' represent the linkages between references and services. Thus, the building of a business solution can be progressed as follows:

1. *Component building*: the implementation of components which provide services and consume other services
2. *Module building*: Components are assembled together as modules; Modules are deployed within an SCA System. An SCA System represents a set of services providing an area of business functionality within a single organization (e.g. inventory control in distributor '*i*').
3. *Subsystem building*: To help build and configure an SCA system, subsystems are used to group and configure related modules. Subsystems contain '*Module Components*', which are configured instances of modules. Subsystems, like modules, also have Entry Points and

External Services. Subsystems can also contain Wires that connect together the module components, entry points and external services.

4. *System building*: the assembly of subsystems to build the business application through the wiring of service references to services; that is, the subsystems are then linked together to form a cohesive solution.

3.2 Model of the Proposed Distribution Chain

Figure 3 shows the assembly model of the proposed distribution chain. The system consists of four subsystems: the first subsystem is the '*InitSystem*', which is the Entry Point of the system. *InitSystem* has only one module component called '*Initialization*', which loads the necessary enterprise data and the market data through Service Data Objects (Beatty et al, 2003); for brevity, these details are not shown.

The second subsystem is the '*IterativeProcess*' subsystem that is responsible for executing the iterative process; the module component '*HandleIterativeProcess*' of this subsystem realizes the iterative process by starting the iterative process for the first time, get the services of the necessary subsystems (subsystems '*StrategicDecisions*' and '*TacticalDecisions*') during successive iterations, and finally, terminates the iterative process when the iterative process converges.

The third subsystem is the *StrategicDecisions* subsystem that will compute the strategic design values. This subsystem has a component module called 'StrategicBusinessProcess', which is the

realization of the business logic represented by the strategic module in figure 2. The fourth subsystem is the *TacticalDecisions* subsystem which is responsible for computing tactical values; for this purpose, this subsystem get the services of four other module components, such as '*InventoryWholeSDist*' (meaning inventory control at wholesalers and distribution centres), '*InventoryRetailers*' (inventory control at retailers), '*TransportWholeSDist*' (transport scheduling from wholesalers to distribution centres), and '*TransportDistRetail*' (transport scheduling from distribution centres to retailers). The subsystem *TacticalDecisions* gets the services of the four other modules locally. However, another design may place these four module components as remote service references to *TacticalDecisions*.

3.3 Advantages of the Service-Oriented Implementation

The main reason for using SCA for implementing modules is to achieve visibility across the distribution chain. In addition to visibility, using SCA offers many other benefits too such as:

- *Conquering complexity*: the assembly model greatly reduces complexity associated with developing large business applications by providing a modular way to unify services provided by the application. The subsystems can be run on different machines, maintained and updated by different enterprises. New components can be added or existing components can be updated in an incremental way at runtime.
- *Structured line of business offers agility*: By structuring applications as a series of services, IT assets become more agile and enterprises are better able to adapt to the dynamic business environments. Components are coded with distinctive mathematics so that it will make them easy to adapt to new and changing requirements - this is a fundamental issue in the modern

distribution chain; the distribution chain must be continuously adapting to changing market conditions.

- *Integration with ease*: In principle, every component is reusable independent of its context; this means, a component representing a module should be ready to be used by any other components (modules) running at a remote location. Clients of a component do not need any knowledge of how the component is implemented; they need to know only the interface to the component. As long as the interface remains unchanged, a component can be modified without affecting the clients (Beatty et al, 2003; Kozaczynski, 1999).

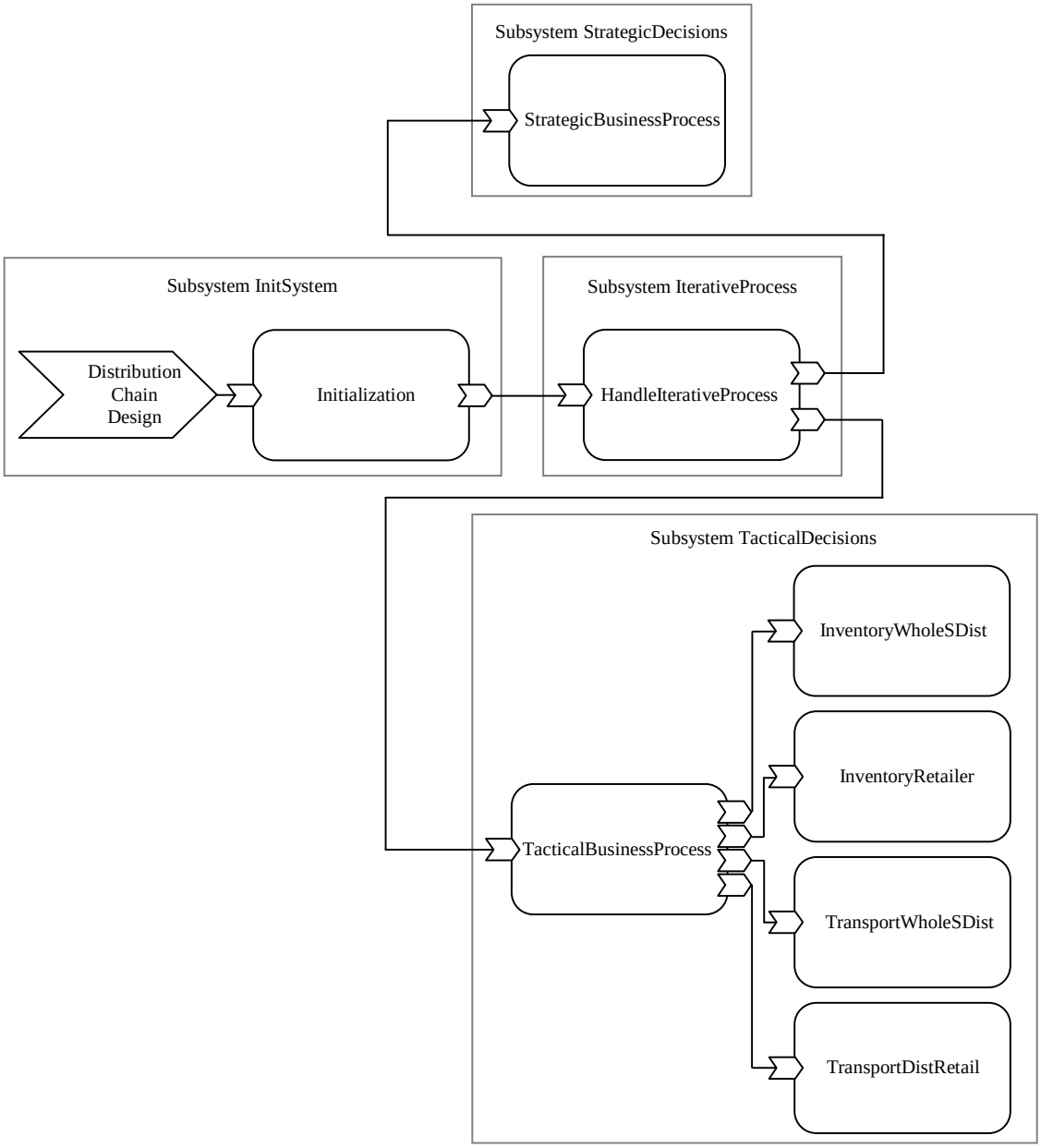

Figure 3: The System Assembly

4. PERFORMANCE ANALYSIS

The previous two sections explain how the design and implementation of the proposed distribution chain achieves variability and visibility, the two objectives of an adaptive distribution chain. This section deals with the third objective, namely velocity. In this section, a performance analysis is done in order to investigate whether the information flow across the proposed distribution chain is fast enough.

4.1 Scenario Description

A distributor wants to know how its inventory level influences the distribution chain. First, the distribution chain will process the input data by feeding it to the strategic module and then to the tactical modules and sub-modules. The strategic module decides whether the input data adheres to the strategic goals; the tactical modules make sure that the input data adheres to the tactical goals too.

Thus, when a collaborator (*or client*, in client-server paradigm) sends details (*a request*) to the distribution chain (*server*), it gets the *response* after going through the strategic module and the tactical modules, repetitively (iterations). The number of iterations depends on the input data and therefore not pre-fixed. Thus, the response time is not constant.

The response time is a very important design parameter that determines how the interface between the collaborator and the distribution chain can be implemented. For example, if the response time is in the order of tens of seconds meaning velocity is not fast enough, then the

interface should be realized as a non-blocking asynchronous messaging service. On the other hand, if the response time is in the order of seconds, then blocking and synchronous Web service is the best option for realizing the interface.

4.2 Brief Overview of the Choices of Modelling Tools

A mathematical model of the distribution chain is needed to measure the response time. Generally, Automata, Sateflow, and Petri nets, are the tools that are most suitable for modelling distributed systems, e.g. the distribution chain; interested reader is referred to Davidrajuh and Molnar (2006) for a short survey of tools. In this paper, a Petri net tool called General Purpose Petri Net Simulator (GPenSIM, 2006) is used for the modelling. GPenSIM is a non-graphic program written in MATLAB.

4.3 Performance Analysis

To calculate the response time from the distribution chain to the distributor, first the distribution chain should be mathematically modelled, so that simulations can be run on the model. The following steps are involved in the modelling and simulation:

1. Creating a Petri net (mathematical) model of the distribution chain
2. Creating a simulation program for the model
3. Running the program (simulations)

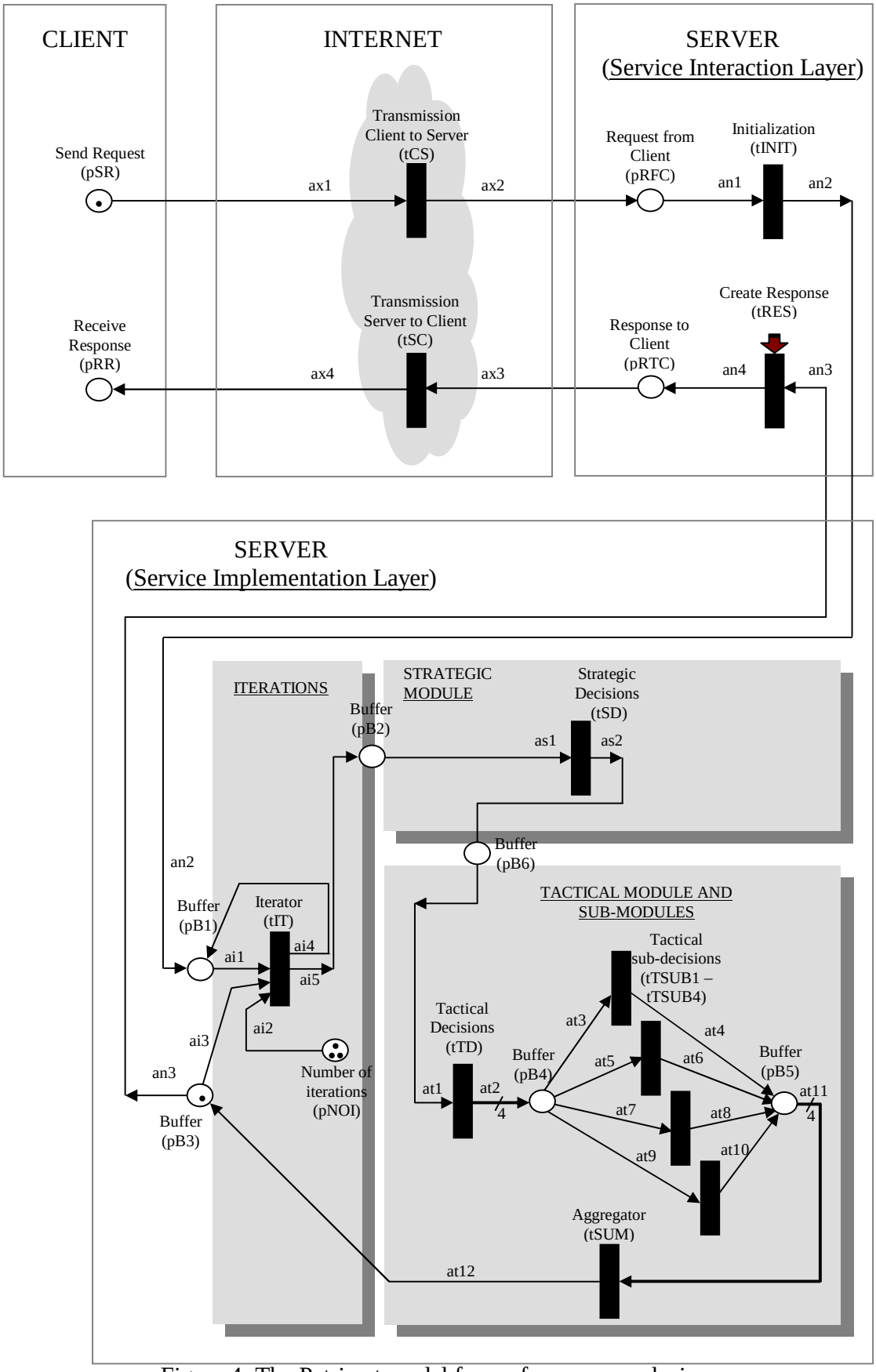

Figure 4: The Petri net model for performance analysis

4.3.1 Mathematical model

The distribution chain that is shown in figure 3 should be converted into a Petri net model. The Petri net model is shown in figure 4. It is relatively easy to convert a discrete event based system like the distribution chain shown in figure 3 into a Petri net model: all the processing nodes are replaced by transitions with an input and an output place. Interested readers are referred to the standard textbooks on Petri nets, for example, Cassandras and LaFortune (1999).

4.3.2 Simulation Program

The Petri net model shown in figure 4 is graphical and is drawn on a piece of paper. The information shown in the figure should be converted into a set of instructions using a Petri net toolbox like GPenSIM. The simulation program written in GPenSIM is given in the appendix.

4.3.3 Running Simulations

Table 1 shows the data used for simulations.

Event	**Name of the transition in figure 4.**	**Value (in milliseconds)**
Client to Server transmission time	tCS	Normal distribution (2000,50)
Server to Client transmission time	tSC	Normal distribution (2000,50)
Initialisation time	tINIT	Uniform distribution (280,320)
Time for packing the results	tRES	Uniform distribution (1, 10)
Time for making strategic decisions	tSD	Uniform distribution (80, 100)
Time for making tactical decisions	tTD	Uniform distribution (25, 35)
Time for making tactical sub-decisions	tSUB1	Uniform distribution (10, 15)
Time for making tactical sub-decisions	tSUB2	Uniform distribution (10, 15)

Time for making tactical sub-decisions	tSUB3	Uniform distribution (10, 15)
Time for making tactical sub-decisions	tSUB4	Uniform distribution (10, 15)
Time for summing the results	tSUM	0
Time for starting next iteration	tIT	0

Table1: Time (in milliseconds) for different events

In addition, the number of iterations (NOI) was taken as 3.

Simulation result indicates that the client (distributor) gets the response in 4.6 seconds; this means, the velocity of information flow is acceptable. This also means that the interface between the distributor and the distribution chain could be realized either as a Web service.

5. CONCLUDING REMARKS

The distribution chain presented in this paper achieves all the three objectives of an adaptive chain, namely variability, visibility, and velocity.

To achieve variability, the design approach uses iterations to make the individual modules adaptive to the quickly changing market conditions. Diving a distribution chain into a number of independent modules paves a distinctive advantage: Due to the geographically dispersed nature

of the collaborators, the individual modules of the distribution chain can be executed independently in different places. Since the modules are going to be executed independently by different collaborators in different places, maintenance and further improvements of these modules can also be done independently. Thus, there is no need to design all the modules with the same type of mathematics. The most suitable algorithm and proper mathematics can be used for realizing each module.

To achieve visibility, this paper proposes Web services based Service Component Architecture (SCA) for implementation of the distribution chain. As given in section 3, this implementation provides a number of advantages like conquering complexity, agility due to structured line of business, and ease of integration. In addition, SCA offers many other benefits too, like separation of business logic from infrastructure capabilities (infrastructure capabilities like Security, Transactions, etc.), programming language independency by separation of service interfaces from service implementation, diverse invocation mechanism (Web services, Messaging, etc.), diverse programming styles (asynchronous message-oriented, synchronous Remote Procedure Call), etc.; Beatty et al (2003) and Beisiegel et al (2005) give an overview of SCA and its benefits.

To analyse the speed of information flow the distribution chain (velocity), this paper proposes use of GPenSIM, a Petri net-based tool for modelling and simulation of discrete event systems. The methodology used for performance analysis is elegant, powerful yet simple.

Limitations of this work: This paper assumes that when the market conditions change, new values for parameters such as demand, transportation costs, etc. can trigger a new round of iterations to fine-tune the modules of the distribution chain. Firstly, this work did not find out the most important parameters that can represent the change of market conditions. Secondly, this work does not provide the necessary the mathematical equations for processing the parameters that represent the changes in the market.

APPENDIX: THE SOURCE FILES

There are three source files involved:

1. The main file for simulation: ‘adaptivechain.m’
2. Petri net definition file: ‘chain_impl.m’, and
3. Definition file for transition tRES: ‘tRES_impl.m’

A.1 Main file: adaptivechain.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%      File: adaptivechain.m
%%      The main simulation file.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
PetriNet = build('chain_def');
NOI = round(unifrnd(2,4)) % number of iterations
sources = {'pSR',1, 'pNOI',NOI, 'pB3',1};
SimulationRESULTS = gpensim(PetriNet, sources);
printsys(PetriNet, SimulationRESULTS);
```

A.2 Petri net definition file: chain_def.m

```
function [PN_name, set_of_places, set_of_trans, set_of_arcs]...
    = chain_def()
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% File: chain_def.m: Definition of the Petri Net model of the
%%                       adaptive distribution chain
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
PN_name='Petri Net Model of Adaptive Supply Chain';

set_of_places={'pSR', 'pRR', 'pRFC', 'pRTC', 'pNOI', ...
    'pB1', 'pB2', 'pB3', 'pB4', 'pB5', 'pB6'};

set_of_trans={'tCS','normrnd(2000,50)', 'tSC','normrnd(2000,50)',...
    'tINIT','unifrnd(280,320)', 'tIT',0,...
    'tRES','unifrnd(1, 10)', 'tSD','unifrnd(80, 100)',...
    'tTD','unifrnd(25, 35)', 'tSUB1','unifrnd(10, 15)',...
    'tSUB2','unifrnd(10, 15)', 'tSUB3','unifrnd(10, 15)',...
    'tSUB4','unifrnd(10, 15)', 'tSUM',0};

set_of_arcs={'pSR','tCS',1, 'tCS','pRFC',1, 'pRTC','tSC',1,...
    'tSC','pRR',1, 'pRFC','tINIT',1, 'tINIT','pB1',1,...
    'pB1','tIT',1, 'pNOI','tIT',1, 'pB3','tIT',1,...
    'tIT','pB1',1, 'tIT','pB2',1, 'pB3','tRES',1,...
    'tRES','pRTC',1, 'pB2','tSD',1, 'tSD','pB6',1,...
    'pB6','tTD',1, 'tTD','pB4',4, 'pB4','tSUB1',1,...
    'pB4','tSUB2',1, 'pB4','tSUB3',1, 'pB4','tSUB4',1,...
    'tSUB1','pB5',1, 'tSUB2','pB5',1, 'tSUB3','pB5',1,...
    'tSUB4','pB5',1, 'pB5','tSUM',4, 'tSUM','pB3',1};
```

A.3 Definition file for transition tRES: tRES_def.m

```
function fire = tRES_def (PN)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%  File: tRES.m
%%      This file defines the transitions tRES
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
p1 = get_place(PN, 'pNOI');
fire =  (p1.tokens == 0);
```

REFERENCES

Beisiegel, M., Blohm, H., Booz, D., Dubray, J.-J., Edwards, M., Flood, B., Ge, B., Hurley, O., Kearns, D., Lehmann, M., Marino, J., Nally, M., Pavlik, G., Rowley, M., Sakala, A., Sharp, C., and Tam, K. (2005). SCA Service Component Architecture: Assembly Model Specification. SCA Version 0.9, November 2005.

Beatty, J., Brodsky, S., Nally, M., and Patel, R. (2003). Next-Generation Data Programming: Service Data Objects: A Joint Whitepaper with IBM and BEA. BEA Systems, Inc. and International Business Machines Corp, November 2003.

Caputo, A., Cucchiella, F., Fratocchi, L., Pelagagge, P., and Scacchia, F. (2004). Analysis and evaluation of e-supply chain performances. *Industrial Management & Data Systems, 104 (7), 546-557*

Cassandras, G. and LaFortune, S. (1999). Introduction to Discrete Event Systems. Hague: Kluwer Academic Publications

Childerhouse, P., Hermiz, R., Mason-Jones, R., Popp, A., and Towill, D. (2003). Information flow in automotive supply chains – identifying and learning to overcome barriers to change. *Industrial Management & Data Systems, 103 (7), 491-502*

Davidrajuh, R. (2000). Automating Supplier Selection Procedures. PhD thesis, Norwegian Univ. Science and Technology (NTNU), ISBN-82-7984-159-8

Davidrajuh, R. and Ma, H. (2006). Developing a Modern Distribution Chain: A Three-Pronged Approach. Proceedings of the IEEE International Conference on Service Operations and Logistics, and Informatics, 21-23, June 2006, Shanghai, China

Davidrajuh, R. and Molnar, I. (2006). A Tool for Teaching Mathematical Modeling to Information Systems Students, Scheduled for publication in Issues in Information Systems

GPenSIM (2006). (Accessed July 01, 2006) http://ilab16.ilab.ux.his.no/GPenSIM/

i2 (2004) i2: The Foundation for Supply Chain Optimization. i2 inc.

Kozaczynski, W. (1999) “Composite nature of components”, 1999 International Workshop on Component-Based Software Engineering, http://www.sei.cmu.edu/cbs/icse99/papers, 1999

Ma, H. Z. and Davidrajuh, R. (2005). An iterative approach for distribution chain design in agile virtual environment. *Industrial Management & Data Systems, 105 (6), 815-834*

MATLAB (2006). (Accessed February 21, 2006) http://www.mathworks.com

Petri net world (2005). (Accessed July 10, 2005) http://www.daimi.au.dk/CPnets/

Rahman, Z. (2004). Use of internet in supply chain management: a study of Indian companies. *Industrial Management & Data Systems, 104 (1), 31-41*

SAP (2002). Adaptive Supply Chain Networks. SAP White Paper

Smith, A. (2006) Supply chain management using electronic reverse auctions: a multi-firm case study. *International Journal of Services and Standards 2006 - Vol. 2, No.2 pp. 176 - 189*

Shih, D., Huang, S. and Lin, B. (2006) Linking secure reverse auction with web service. *International Journal of Services and Standards 2006 - Vol. 2, No.1 pp. 15 – 31*

Tijms, H. (1994). Stochastic Models: An Algorithmic Approach. New York: John Wiley & Sons

Wu, W., Chiag, C., Wu, Y., and Tu, H. (2004). The influencing factors of commitment and business integration on supply chain management. *Industrial Management & Data Systems, 104 (4), 322-333*

Wang, A. and Qian, K. (2005). Component-Oriented Programming. Wiley-Interscience: John Wiley & Sons, 2005

Wei, J., Platt, R., White, B. and Jasquith, A. (2006). Development of standardised e-business solutions via e-chain analysis in the digital utility. *International Journal of Services and Standards 2006 - Vol. 2, No.2 pp. 117 – 136*

World Trade (2004) “Adadptive Supply Chain” Mar 2004, 17 (3), 41-41