Large Scale Order Processing
through Navigation Plan Concept

João Eduardo Ferreira¹, Osvaldo K. Takai¹, Kelly R. Braghetto¹, and Calton Pu²

¹ Department of Computer Science, University of São Paulo
Rua do Matão, 1010 - Cidade Universitária
05508-090 São Paulo - SP - Brasil
e-mail: {jef1takai}@ime.usp.br

² College of Computing, Georgia Institute of Technology
801 Atlantic Drive, Atlanta GA 30332-0280- USA
e-mail: calton@cc.gatech.edu

Abstract. Order processing is an important application in electronic commerce and cooperative information systems. On the conceptual side, business process management helps model such applications. On the practical side, workflow systems help implement such systems. We propose the Navigation Plan concept for order processing. On the conceptual side, business steps in order processing are mapped into process algebra and composed into navigation plans. On the practical side, navigation plans are directly executed in the RiverFish Architecture, thus guaranteeing the properties predicted by process algebra. Thus a well-defined order processing application is implemented into a reliable cooperative information system. A widely used application (the DECA system for requesting business TaxID involving several government agencies) demonstrates the usefulness of the Navigation Plan concept in practice.

1 Introduction

Order processing is an important part of electronic commerce. For example, online shopping has been reported to exceed $150B in global e-commerce sales for 2004
Order processing is also an important example of cooperative information systems, since typically it involves several important business processes such as supply chain management, inventory control, and cash flow. Furthermore, typically these processes involve several companies specialized in their function (e.g., manufacturing, transportation, and financial services), making interoperation and cooperation even more explicit. Two of the fundamental challenges in the design and implementation of order processing systems are: (1) interoperation and integration of heterogeneous and autonomous information systems, and (2) flexible evolution and composition of software components that implement the workflow of order processing.

To address these challenges, we analyze and divide an order transaction into three stages. First, a client (either human user or another program) generates an order, i.e., a request for information, physical goods, or services. Second, the order is validated by going through a series of business steps. This is important since legitimate orders can be fulfilled only when they satisfy the requirements specified in those business steps. Third, a validated order is passed along to an appropriate execution process to carry out the request, usually the delivery of requested information, goods, or services in exchange for an appropriate payment.

The main hypothesis of this paper is that the three stages of order processing can be separated, implemented, integrated, and executed through well-defined interfaces that support both the integration of heterogeneous and autonomous information systems and flexible evolution of workflow. Our approach is called Navigation Plan which carries an order transaction through all the four stages in a clean and efficient way. The Navigation Plan concept has been implemented in the RiverFish Architecture (Section 3). RiverFish forms the backbone of the DECA application [6], used by the State of Sao Paulo (Brazil) to register and give TaxIDs to companies being created. DECA has been used by more than 800,000 companies since 2001.

The main innovation of the paper is the Navigation Plan approach’s ability to link a semi-formal description using process algebra [4] [5] [8] to a practical execution environment. On the formal side, the business steps are mapped to process algebra for analysis and composition. On the practical side, navigation plans are directly executed in the RiverFish Architecture [7], which guarantees the properties predicted by the process algebra. The advantages of this link are demonstrated in practice by DECA, a cooperative information system integrating several autonomous components.

The rest of the paper is organized as follows. Related work is outlined in Section 2. In Section 3, a summary of RiverFish Architecture is presented. Section 4 describes the mapping of navigation plans into a semi-formal description based on process algebra. Section 5 discusses the advantages of this formal mapping and the practical mapping to RiverFish in DECA. Section 6 concludes the paper.
2 Related Work

There have been several initiatives to solve the representation and execution of business rules through workflow language and business process management approaches [1] [3] [10]. Workflow Management Coalition (WFMC) defines workflow as: “The automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules.” [11]. A Workflow Management System (WFMS) is defined as: “A system that defines creates and manages the execution of workflows through the use of software, running on one or more workflow engines, which is able to interpret the process definition, interact with workflow participants and, where required, invoke the use of IT tools and applications.” [11]. According to [2], both definitions emphasize the focus on enactment, i.e., the use of software to support the execution of operational processes. In the last couple of years, many researchers and practitioners started to realize that the traditional focus on enactment may be too restrictive.

Another relevant area is Business Process Management (BPM) [2], defined as: “Supporting business processes using methods, techniques, and software to design, enact, control, and analyze operational processes involving humans, organizations, applications, documents and other sources of information”. This definition restricts BPM to operational processes, i.e., processes at the strategic level or processes that cannot be made explicit are excluded. Note that systems supporting BPM need to be “process aware”, i.e., without information about the operational processes at hand little support is possible. These BPM approaches focus on the modeling of business rule processing and constraints. There are definition language initiatives such as [1] [3] [10] to control business rule processing. The mapping of business processes to workflow-based automated execution has been an active area of research. For example, Product-Driven Case Handling (PDCH) approach [3] focuses on a business objective as a ‘product’ with structure and a current state. In contrast to existing workflow systems, the logistical state of the case is not determined by the control-flow status but by the presence of data objects. In PDCH approach, forms are the only way of representing and linking data objects.

Complementing the workflow research, which focuses primarily on enactment, and BPM work, which focuses mainly on modeling, we would like to link a formal model directly to enactment. Our solution is the Navigation Plan concept, which captures and models business steps using process algebra (Section 4) and is directly executed in the RiverFish architecture by a workflow engine (Section 3).
3 Summary of RiverFish Architecture

The RiverFish architecture for management of business processes has been introduced in [6] [7]. In this section, we summarize the RiverFish architecture to make this paper self-contained. There are two types of business process: atomic and composed (i.e. business process composed by others business process). Each atomic business process is described through a set business steps. A business step can be a single action or check point. Business process is initiated through an order. Check points implement rules, validations and constraints. Single actions and check points are used for the manipulation of data. The figure 1 shows these premises with ER-X notation.

RiverFish architecture is designed to represent order processing by modeling and executing business steps to achieve business objectives. RiverFish consists of three modules: unified control, execution of the navigation plan instances, and storing of data. Initially the unified control receives order. In this section, we outline the RiverFish architecture to make this paper self-contained. Further details of the architecture can be found in [7].

The unified control manages three databases. The database called db-AC (database for the Access Control) stores all the information of users, their profiles and access permissions. The db-NP (database for Navigation Plans) stores the business steps for each data set pertaining to order.

The database db-AI, database for Arrived data and navigation plan Instances, stores order, its data and Navigation Plan instances.
The Navigation Plan maps all business steps according to data belonging to all order of a business objective. The aim of the Navigation Plan is to link the order to the respective business steps.

Figure 2a and 2b illustrates Riverfish description using the UML notation [9]. The system is initiated upon the receiving of order(s). Firstly, the unified control identifies, certifies and authenticates the order using the access control database (db-AC) according to item 1 and 1.1 of UML diagram on figures 2a and 2b. Secondly, the Navigation Plan Instance is assigned to the order through the monitor service of Navigation Plan Instance called instance generator (item 1.1.1 and 1.2).

This service asks the db-NP what kind of Navigation Plan should be assigned. The db-NP then responds by assigning the Navigation Plan instance (item 1.2). The order and its Navigation Plan instance are stored in db-AI.

After the unified control module has finished the Navigation Plan Instance, RiverFish activates the execution module so as to process the Navigation Plan Instances (item 2 and 3). Such monitor is a process scheduler capable of interpreting and executing the steps of the instanced Navigation Plan.

The data quality pertinent to the order in question increases according to the successful execution of business steps. With respect to each step of the Navigation Plan Instance, RiverFish provides a process scheduler capable of interpreting and executing the steps of the instanced Navigation Plan.
Plan instance which has been successfully executed, means that the order is getting to its final destiny. The steps and their results are stored in the database \textit{db-AI}. The order as well as its Navigation Plan instance must satisfy a certain degree of quality before being sent to the database systems.

The last module denominated as the storing of data is implemented by the database monitor and quality checker. The quality checker monitors the order until the desired quality is accomplished. This monitoring is done through intermittent queries to the Navigation Plan instance. The moment that the quality is achieved, the stored data in database \textit{db-AI} is transferred to the respective database systems. This data transference task is done by the database monitor (item 4, 5, 6 and 7).

![Figure 2b: Details of Riverfish architecture components through UML notation.](image)

4 Semi-Formal Description of Navigation Plan

We use a simplified approach of working permit to illustrate the semi-formal description of Navigation Plan. The working permit is an order request belongs to the application for DECA (business objective) [6].
4.1 Simplified description of work permit order

As illustrated in figure 1 the mains premises of RiverFish architecture are below explained with DECA environment application.
1 - All electronic services are available on the Web and classified in business objectives. Examples of business objective are application for: DECA work permit, GIA tax control, AIDF printed form control and so forth.

2 - The business objectives are made up of business processes. Example of business processes are: working permit (get_DECA), changing partners (ch_partners), changing tax collection classification (ch_tax_colec_class) and so forth.

3 - Business process can be associated to generate composite process.

4 - Each business process is described through a set business steps.

5 - A business step can be a single action or check point.

6 - Business process is initiated through an order.

7 - Check points implement rules, validations and constraints.

8 - Single actions and check points are used for the manipulation of data.

The business process called get_DECA was chosen for illustrating the semi-formal description of Navigation Plan. The steps of “get_DECA” were simplified and described according to business steps below.

- p1: access form get_DECA.
- p2: fill out get_DECA form.
- p3: send get_DECA form.
- p4: run the data validation at State Government.
- p5: run the data validation at Federal Government.
Steps p1, p2, and p3 are directly transformed to actions a1, a2 and a3 according to Theory of Process Algebra [4] [5] [8]. However steps p4 and p5 are check points, and they need to be transformed to actions through Function F, according to our definition 1.

**Definition 1:** Let’s define F as the set of the functions that transform check points to actions according to Theory of Process Algebra. As such, F = \{ F_1, F_2, ..., F_m \}, where, each F_i transforms the check point p_i to actions a_i, a_{i+1}, ..., a_n. If S is a set of check points and A is a set of actions, then S is a domain and A is the image of F.

\[ F_i : S \rightarrow A \]

For the specific case of steps p4 and p5:

\[ F_4(p4) = \{ \text{not}p4, \text{ok}p4 \} \] \[ F_5(p5) = \{ \text{not}p5, \text{ok}p5 \} \]

Where:

- \text{not}p4 means that State Government does not support the check point, i.e., the return of check point execution is false.
- \text{ok}p4 means that State Government supports the check point, i.e., the return of check point execution is true.
- \text{not}p5 means that Federal Government does not support the check point.
- \text{ok}p5 means that Federal Government supports the check point.

The action names are changed in order to homogenize semi-formal notation: \text{not}p4 to a4, \text{ok}p4 to a5, \text{not}p5 to a6 and \text{ok}p5 to a7.

Observe that F_i necessarily needs not be a binary function. This binary function could generate n possible actions for a determined step. In this example, only two actions were enough to represent the check point situation: to attend to or not.

In order to present the real situation the generic action a_g was defined. In this case, each check point is transformed to a supported check point. This action is executed by a “supreme” user that is capable of transforming any temporary inconsistency check point to a consistency check point. Inconsistency check points are generated when check points are not supported.

Thus, the get_DECA simplified process can be visualized through the graph as illustrated in Figure 3. An equivalent graph can be seen in Figure 4. However the graph of Figure 4 has one state less.
4.2 Navigation Plan concept and Process Algebra

In this research, Process Algebra has been used to represent the semi-formal description of Navigation Plan. The summary of the main Process Algebra concepts
can be seen in definitions 2 through 5. Definitions 6, 7 and 8 are new extensions of
Process Algebra. These new extensions were created and used in this research.

**Definition 2:** $P$ is defined as a set of process.

**Definition 3:** $\Phi \in P$. $\Phi$ is the action that defines a determined final state of process $P$.

**Definition 4:** Let’s define $A$ as set of actions. $A = \{a_1, a_2, \ldots, a_n\}$
Each action $a_i \in A$, $\cdot : P \rightarrow P \cdot$ is a sequential composition operator of process $P$.
Examples:

$P_1: a_1. a_2. \Phi$
$P_2: a_1. a_2. \Phi + a_2. a_3. \Phi$

The $+$ symbol represents the executions of alternative actions of process $P_2$.
There are two possible ways: a) execution of $a1$ and $a2$; or b) execution of $a2$ and $a3$.

**Definition 5:** The states of Process $P_1$ are represented through the sequence of executed actions $a_i$, whereby:

$a_1. a_2. \Phi \rightarrow a_2. \Phi \rightarrow \Phi \rightarrow$ defines the states: $a_1.a_2.\Phi, a_2.\Phi, \Phi$.

Process Algebra details can be seen in references [4] [5] [8]. Using the above definitions, the business process nominated get_DECA is represented as follows.

$P_{get\_DECA} = a_1. a_2. a_3. ( (a_5. a_7. a_6. \Phi + a_6. \Phi) + (a_4. a_7. \Phi + a_6. \Phi) )$

A significant portion of the business process get_DECA is reused in other business processes belonging to application for DECA work permit (business objective). The business process called changing partners ($P_{ch\_partners}$) uses the same business steps $p4$ and $p5$, as already defined in $P_{get\_DECA}$.

Business step $p6$ is added to $P_{get\_DECA}$. Thus, steps belonging to $P_{ch\_partners}$ are:

- $p11$: access form for changing partners.
- $p21$: fill out form for changing partners.
- $p31$: send the form for changing partners.
- $p4$: run the data validation at State Government.
- $p5$: run the data validation at Federal Government.
- $p6$: verify the financial situation.
Analogous to the P\textsubscript{get\_DECA}, steps p11, p21 and p31 are directly transformed into a11, a21 and a31. Step p6 is also a check point. This step needs to be transformed into actions through function F\textsubscript{6}(p6), similar to steps p4 and p5. Thus, F\textsubscript{6}(p6) = \{ notp6, okp6 \}. The set is changed so as to homogenize notation: notp6 to a8, okp6 to a9. F\textsubscript{6}(p6) = \{ a8, a9 \}.

The expression of Process Algebra P\textsubscript{ch\_partners} is as follows.

\[
P_{ch\_partners} = a_{11} . a_{21} . a_{31} . (a_{5} . (a_{7} . (a_{9} . \Phi + a_{8} . ag . \Phi) + a_{6} . ag . (a_{8} . ag + a_{9} . \Phi)) + a_{4} . ag . (a_{7} . (a_{9} . \Phi + a_{8} . ag . \Phi) + a_{6} . ag . (a_{8} . ag + a_{9} . \Phi)) + a_{6} . ag . (a_{8} . ag + a_{9} . \Phi)).
\]

**Definition 6:** The representation of business steps (single actions or check points) belong to the business processes of all business objectives in Information System is denominated as **Navigation Plan**. Single actions are directly transformed into actions according to Process Algebra. Each check point is transformed to action through a function F according to definition 1. The Navigation Plan is described through a graph with \( n \) entries for \( n \) business processes. These business processes share various subgraphs.

Figure 5 shows the Navigation Plan with processes P\textsubscript{get\_DECA} and P\textsubscript{ch\_partners}. Each new business process created is included into the Navigation Plan. Each business process has its own initial and final states which are not necessarily coincident.

**Figure 5:** Representation of the Navigation Plan in RiverFish architecture.

**Definition 7:** The *instance of Navigation Plan* is a representation of a specific order linked with the business steps (single actions and check points). This instance has an
order and its subgraph of Navigation Plan. An instance stores order, its actions and check points which are described in the Navigation Plan. The instantiation concept of the Navigation Plan allows the execution specificity of each order. The instance of Navigation Plan is used in the second part of the order transaction.

Figure 6 illustrates a common subgraph between two business processes: $P_{\text{get\_DECA}}$ and $P_{\text{ch\_partners}}$.

The business process representation is factored in disjoint segmented processes (Pi) as follows.

\[
\begin{align*}
P1 &= a1 . a2 . a3 \\
P2 &= a11 . a21 . a31 \\
P3 &= a5 . (a7 . \Phi + a6 . ag . \Phi) + a4 . ag . (a7 . \Phi + a6 . ag . \Phi) \\
P4 &= a8 . ag . \Phi + a9 . \Phi
\end{align*}
\]

The business processes are now represented by:

\[
\begin{align*}
P_{\text{get\_DECA}} &= P1 . P3 \\
P_{\text{ch\_partners}} &= P2 . P3 . P4
\end{align*}
\]

$P3$ can be factored by $P3' = a7 . \Phi + a6 . ag . \Phi$, generating:

\[
P3 = a5 . P3' + a4 . ag . P3'
\]

![Diagram of business processes](image)

**Figure 6: Factoring business processes in disjoint segment processes.**

A new definition is not necessary for the concept of the disjoint segmented process. The disjoint segmented process can be understood as a temporary state according to definition 5. However, we need to differentiate the action $\Phi$ within $P3'$ in $P_{\text{get\_DECA}}$ and $P_{\text{ch\_partners}}$. 
In $P_{\text{get\_DECA}}$, $\Phi$ generates the final state, but $P_{\text{ch\_partners}}$ has an intermediate state. In order to attend to the differentiation, a new definition 8 is conceived. Definition 8 is the refining of definition 3.

**Definition 8.** $\Omega \in P$. $\Omega$ is the binding action that defines an intermediate or final state belonging to the segment of process $P$. If process segment $P$ is final then $\Omega = \Phi$, otherwise, $\Omega$ represents a binding action of the segment processes. The process $P3'$ can be rewritten using the definition 8, whereby:

$$P3' = a7 \cdot \Omega + a6 \cdot ag \cdot \Omega$$

### 5 Discussion and Implementation

In the previous section, we described the modeling of navigation plans using process algebra. The modeling advantages of using process algebra are: a precise description of each business step and predictable properties when composing the steps into a composite navigation plan. Process algebra also provides the control over temporal and causal dependencies among the business steps. In addition to this proper modeling of navigation plans, process algebra has also facilitated the management of application complexity.

Recall the RiverFish Architecture that implements navigation plans (Section 3) in the DECA application. DECA application is composed of 23 business processes. DECA code application controls 284 rules and 380 check points.

An important technical contribution of the Navigation Plan is the facility to change or evolve the business steps (single action, rules, check points, or constraints). If $P_{\text{get\_DECA}}$ gives the new business step to update the $p4$ step, for example:

new-$p4 =$ run the data validation at city hall Government.

So, the main change is reduced only a step update in Navigation Plan description. It is made through action updates in Navigation Plan. If new-$p4$ is an evolution, i.e., a new business step insertion, then this situation looks like step $p6$ in $P_{\text{ch\_partners}}$.

However, for each step change the Navigation Plan is factored again for getting the new expression of process algebra as illustrated in figure 6. With this new expression the RiverFish’s architecture and its services (instantiation and execution) does not suffer any changes. It means that the changes are easy localized and replaced in Navigation Plan without to affect others business steps and implementation code in RiverFish approach.

Another major technical contribution of the Navigation Plan concept is the implementation of navigation plans in the RiverFish architecture, which translates navigation plans directly into execution through RiverFish’s Navigation Plan Instance.
service. The direct execution of navigation plans guarantees the properties predicted by process algebra. The Navigation Plan Instance service also schedules the execution sequence of business steps according to the dependency specifications and constraints contained in each navigation plan.

The Navigation Plan-based DECA system [7] was introduced into production use in March 2000. By November 2004, DECA was serving more than 800,000 active companies and 900,000 contributor users. The DECA database system (approximately 3GB size) is divided into 84 tables for data storing of companies, users, addresses, tax control parameters, and order (requests and approvals). Compared to the previous system that used manual interference to solve the transactional conflicts of sequence steps, DECA has achieved important service improvements as the system workload increased between March 2000 and November 2004. For example, the number of requests for new business TaxID, when a company starts its activities, has increased by 17% from less than 7,000 to more than 8,000 per month. At the same time, the response time of the DECA system (from the filing of a request to its fulfillment) has decreased by two orders of magnitude, from tens of days to a few hours. The main reason for this reduction in response time is the success of an automated DECA system to execute the check points through navigation plans. By Nov 2004, the percentage of initial business TaxID requests completed without human intervention reached 87%.

6 Conclusion

In this paper, we described the Navigation Plan approach to design and implement large scale order processing in cooperative information systems. On the formal side, the Navigation Plan uses the process algebra to describe and compose business steps. On the practical side, the Navigation Plan is implemented by the RiverFish Architecture [7]. A widely used application for the application of business TaxIDs, called DECA [6], has been implemented on the RiverFish Architecture. DECA demonstrates the advantages of using Navigation Plan concept in practice.

The main contribution of the paper is a detailed description of Navigation Plan concept. On the formal side, the navigation plans are mapped to process algebra. This mapping will allow us to explore interesting algebraic proprieties for order processing such as bisimilarity, recursion, composition, concurrency and parallelism.

On the practical side, the navigation plans are directly executed by RiverFish, which guarantees the properties predicted by process algebra. The production use of the DECA system demonstrates the reliability achieved by a Navigation Plan-based order processing system.
References


