Abstract no. 025-0096

Process-Aware View of the Relationship between Software Architecture and Flexibility Costs

Zhenyu Liu

School of Management, Xiamen University

Bao Xin Li Ying Building, Xiamen University, Xiamen, Fujian, CHINA

e-mail: zhenyliu@xmu.edu.cn

Phone: +865922187020

Sien Chen

School of Management, Xiamen University

Bao Xin Li Ying Building, Xiamen University, Xiamen, Fujian, CHINA

e-mail: sandy80@vip.sina.com

Phone:+865923629687

POMS 23rd Annual Conference

Chicago, Illinois, U.S.A.

April 20 to April 23, 2012

ABSTRACT:

The paper clearly demonstrates the relationship between software architecture and costs of flexibility in the case of process-aware software architecture. Complementing the earlier research studies, we propose an economic model to assess the impact of software architecture flexibility strategies on business process-aware. The proposed relationships are all based on empirical and analytical evidence from large case studies. The result describes that supporting business process patterns with process-aware software architecture mainly depends with dynamic evolution and dynamic refinement components. We seek to specify the value and the cost of software architecture flexibility-to-change in support of business process-aware. Process-aware software architecture costs have to be justified by benefits, in particular, generates benefits by supporting business processes.

Keywords: Costs, Flexibility, Process-Aware Software Architecture, Relationship

1. INTRODUCTION

In today's dynamic business world the economic success of enterprises increasingly depends on its ability to react to changes in its environment in a quick and flexible way (Hammer and Stanton, 1995). Process-aware information systems (PAIS) offer promising perspectives in this respect and are increasingly employed for operationally supporting business processes (Dumas et.al, 2005). Meanwhile to provide effective business process support, flexible PAIS are needed which do not freeze existing business processes, but allow for loosely specified processes, which can be detailed during run-time. PAIS should enable authorized users to flexibly deviate from the predefined processes if required (Section 2.1) and evolve business processes over time.

The emergence of different process support paradigms and the lack of methods for comparing existing change approaches have made it difficult for process-aware software architecture (PASA) engineers to choose the adequate technology (Weber et.al, 2007). This makes it difficult for PASA engineers to assess the maturity and the change capabilities of those technologies, often resulting in wrong decisions and expensive investments.

Naturally, managers often raise the question "research on the relationship between software architecture and costs of flexibility?" Although based on the organization or cross-organization theory, business process reengineering theory, and information systems technology to study the flexible have some literature, but flexible perspective on economics literature is relatively limit (Mutschler et.al, 2007; Gebauer and Schober , 2006; Schober and Gebauer 2011; Dreyfus and Wyner, 2011), in particular in the process-aware situations, whereas the a previous research highlights the need for the concrete measurement of software architecture flexibility to determining the value and the cost of software architecture flexibility. Motivated by this observation, we propose an economic model to assess the impact of software architecture flexibility strategies on business process efficiency, which combinative different flexible strategies on the cost of software architecture research to verify the management inspiration in the case of process-aware software architecture.

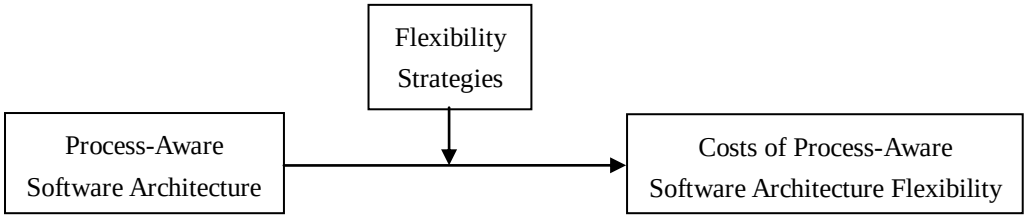

Figure 1. Research on the Relationship between Software Architecture and Costs of Flexibility

The remainder of the paper is structured as follows. A brief theoretical review will be presented below about process-aware software architecture, costs of software architecture flexibility, and costs of software architecture in Section 2. Section 3 describes the theoretical basis of this study. Section 4 describes an economic model to assess the impact of software architecture flexibility strategies on business process-aware. Evidencing relationship between software architecture and costs of software architecture flexibility, besides computing the value and the cost of software architecture flexibility for deterministic process loads is described by Section 5. Finally, Section 6 concludes the paper and outlines an agenda for future research.

2. LITERATURE REVIEW

Companies increasingly adopt PAIS, which offer promising perspectives for more flexible enterprise computing. PAIS are the core of an ongoing trend, and it has drawn the attention of information systems engineers and managers shift from data and objects to the processes that the information system and the intra-organizational or cross-organizational environment in which it operates—is intended to enact, enable or support (Aalst, 2007). Intra-organizational or cross-organizational PASA is different software architecture and components. In this paper, we assume that they are similar to each other. We subsume such different information systems with a process focus on the basis of PASA, and then research on relationship between PASA and costs of PASA flexibility.

2.1 The Process-Aware Software Architecture

"Software architecture (SA) = components + interaction = important decision set" (Wen, 2007).A system is a combination of components that function as a complete whole. One way

to describe a system is through its architecture. Upon the initiation of PASA, decision makers typically have a choice between different levels of flexibility, which is the extent to which the SA can be slowly modified or upgraded during its subsequent lifetime.

The information systems engineering community has concentrated on PASA analysis, design and implementation, using for example workflow technology (Reijers, 2003), service-oriented architectures (Erl, 2005), case handling systems (Aalst and Weske, 2005), or business process management systems (P.Dadam, et al., 2009). The systems management community has focused on process redesign practices (Reijers et.al, 2005), process modeling methods (Recker, 2010; Recker, et.al, 2011), and the impact of process-aware software architecture technology to support process-oriented organizations (Weber et.al, 2010, Adams et.al, 2011), or organizational change to enable process improvement and the management of cultural (Brocke et.al, 2011).

Integration of related technologies and business functions of the components are the parts of a system that the software developers implement in computer code; architecture describes the overall form of the system (Dreyfus and Wyner, 2011). In this paper, PASA is a description of components and dependencies. Through of components and dependencies, PASA is to support different business processes patterns.

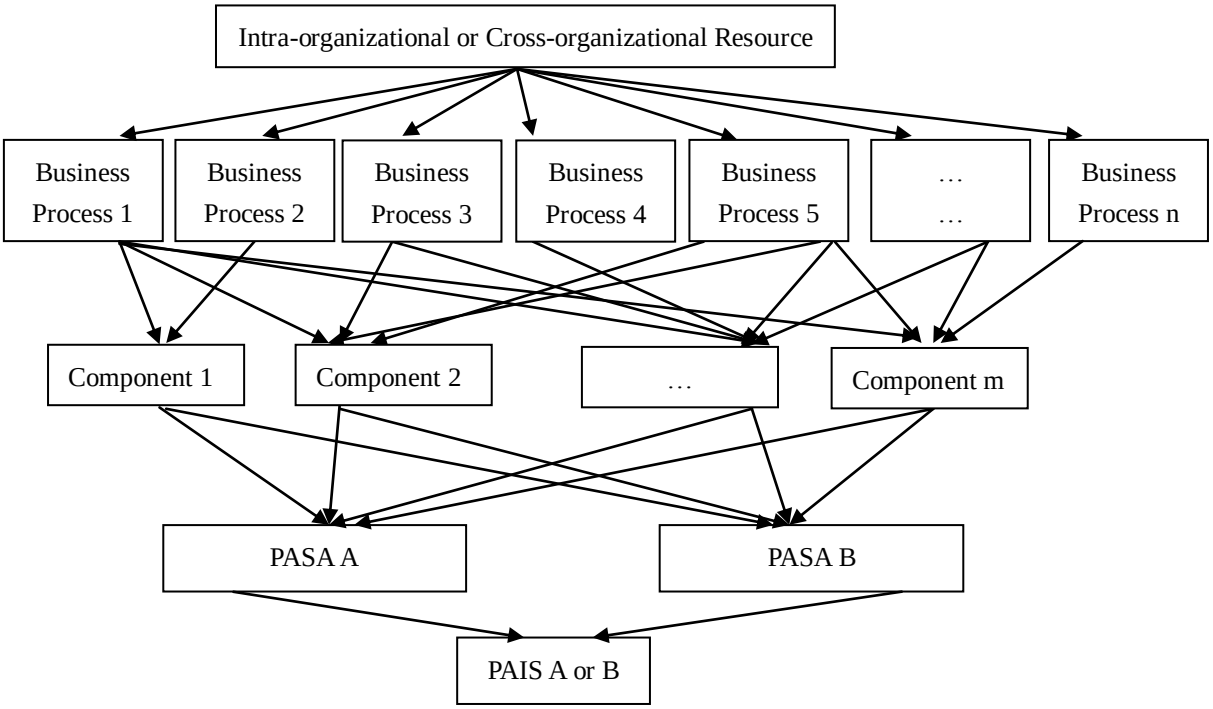

Figure 2. The Relationship between Business Processes and Software Architecture

Pattern Description

A pattern-based analysis (Weber et.al, 2008; Weber et.al, 2005b) combines self-adaption (Li, 2008; Anoint, 2003) and self-evolution (Song, 2011) theory in PAIS, we adopt a set of changes patterns, change support components and dependencies to put forwards PASA according to the situation of the needs business processer facing and changeable environment. To realize business process flexibility, there are 18 characteristic relevant patterns for control flow changes in PASA (Weber et.al, 2008; Weber et.al, 2005b). All these change patterns constitute solutions for realizing commonly occurring changes in PASA which divide the change patterns into adaptation patterns and patterns for changes in predefined regions. Thereby, adaptation patterns support structural process adaptations, whereas patterns for changes in predefined regions allow for built-in flexibility (Weber et.al, 2008).

Support Components and Dependencies Description

A pattern-based analysis combines self-adapting and self-evolution theory in PAIS that is the support for business process-aware feature, mainly including the evolution of the dynamic evolution or dynamic change, dynamic refinement (Aalst et.al, 2002; Reichert et.al, 2003; Li et.al, 2010). The dynamic process of evolution is mainly based on demand and changes in the environment, the use of the evolution operator allow the process model to be modified, making the process model from a version upgrade to another version of the process. In addition, it means that when the process model is changed, the corresponding changes in the dynamic process model then spreads to the running process on the process instance. The dynamic refinement mainly use patterns for changes in predefined regions allow for dealing better with uncertainty by deferring decisions regarding the exact control-flow to run-time. Instead of requiring a process model to be fully specified prior to execution, parts of the model can remain unspecified. Therefore, the supporting business process patterns with process-aware software architecture mainly depends with dynamic evolution and dynamic refinement components.

2.2 The Relationship between Mechanism of Generation Costs and Software Architecture

Responding to business and technology changes often involves modifying the SA and the speed and cost of making changes to the SA is a measure of the system's flexibility. We define the architectural operations that could be performed on a software architecture component as deploy, upgrade, replace, decommission, and integrate. A component is integrated when modifications are made to it that enables it to 'talk' to another component, which creates a dependency. Dreyfus and Wyner (2011) develop measures of architectural

and component complexity and hypothesizes that these constructs affect one dimension of software architecture flexibility: architectural flexibility, the general conclusions from the research are that both complexity at the component level and complexity at the architectural level affect software architecture flexibility. Complexity increases the cognitive efforts required of those making changes to the components and increases the difficulty of finding high-value modifications. Mutschler et.al (2007) present an approach to investigate the complex cost structures of PASA engineering projects based on evaluation models. Because of complexity it is difficult to use in reality.

We also consider the component level on relationship between SA and business process, and document the components and dependencies as observed in that SA as it has come to exist in the cross-organizational environment. Technology skills means the variety of skills and attitudes of the IS staff (Chen et al., 2011). We assume the IS staff who modeling, analysis, design and implementation of PASA responsible for each component have the same technology skills, the complexity of each component is not the same.

Software Architecture Flexibility

Gebauer and Schober (2006) define flexibility-to-use as "the range of process requirements supported by the information system without requiring a major change of the information system", and associate flexibility-to-change with investments required to change, upgrade, or expand an information system. Dreyfus and Wyner (2011), Gebauer and Schober (2006) define these in the context of standards and components, respectively; they apply to SA and information systems as well.

Software architecture flexibility-to-change and flexibility-to-use include modularity and

integration requirements. Modularity as provided by the use of reusable software modules, vendor-independent database connectivity, and object-oriented development tools (Byrd & Turner, 2000; Duncan, 1995). The integration of system functionality, scope of underlying database, user interface and processing capacity provided by software architecture with compatibility of application across platforms (Byrd & Turner, 2000; Duncan, 1995). Related research extends the relationship between software architecture and software architecture flexibility (Table 1).

Table 1. Software architecture flexibility (modified from Chen et al., 2011)

Constructs	Dimensions	Conceptual definitions	Previous studies
Software architecture flexibility-to-use	System functionality	The different basic features of software architecture provides to the users.	(Soh et al., 2000)
	Scope of underlying database	The scope of the database support for deploying reports and analyses for decision making.	(Soh et al., 2000; Chen et al., 2011)
	User interface	The different features and methods software architecture provides to users for interaction.	(Soh et al., 2000; Chen et al., 2011)
	Processing capacity	The capacity provided by software architecture before major performance losses are experienced.	(Gebauer and Schober, 2006; Chen et al., 2011)
Software architecture flexibility-to-change	Dynamic evolution and Dynamic refinement	Supporting 18 characteristic relevant patterns for control flow changes. The extent to which software architecture components can be added, dropped, linked, and replaced without incurring high transition penalties.	(Dreyfus and Wyner, 2011; Chen et al.,2011; Byrd and Turner, 2000; Duncan, 1995)
No usage of software architecture	Not processed by the software architecture	Process tasks performed manually outside of the software architecture in question. For example, the method can take human	(Gebauer and Schober, 2006; Zhang, 2009)

	under consideration	intervention to perform a new workflow task, or part of the process will be outsourced to a third party organization, or no usage of software architecture.	
Component flexibility-to-change	Supported by dynamic evolution and dynamic refinement components	The number of variety of modifications an IT worker can implement on a software component without incurring significant cost or performance penalties.	(Dreyfus and Wyner, 2011)
Component flexibility-to-use	Supported by system functionality and database and user interface and processing capacity components	The extent to which the data in an application can be accessed and modified by interactive users, batch commands, and programmatic access for different purposes without high transition or performance penalties	Flexibility in use (Gebauer et al, 2006; Dreyfus and Wyner, 2011)

Software Architecture Cost

Dreyfus and Wyner (2011) have argued that the complexity of this architecture has an impact on the ease with which designers are able to modify that architecture. Software architectural flexibility cost mainly includes the cost of making an architectural change. Dreyfus and Wyner (2011) defined relevant approach as:

COST = β0 + β1 COMPTIME + β2 DATA_COMP + β3 CLOSE_W + ui",（ui - random disturbance; β0, β1, β2, β3 – parameters）

The respondent's time with the component, COMPTIME, is positively associated with COST; Data complexity, DATA_COMP, because it is not hidden from the components that

access the data, has a significant effect on flexibility; Closeness has a significant effect on flexibility, CLOSE_W, is to all other components, not just to those that integrate with it directly. Based on this method can be used to calculate the cost of SA changes.

A previous research develop a theory to determine the impact of IS flexibility on the cost efficiency of business processes (Mutschler et.al, 2007; Gebauer and Schober , 2006; Schober and Gebauer 2011), we extend a previous theory of software architecture flexibility, concreting consideration of dynamic aspects, including software architecture life time, software architecture type, concreting operationalizing the weights of various software architecture flexibility components. Schober and Gebauer (2011) find real option analysis (ROA) can overestimate its value, in particular in low-risk situations, whereas the deterministic treatment of IS flexibility tends to underestimate its value. In this paper, we use the deterministic treatment of software architecture flexibility tends to estimate its value.

3. MATERIAL AND METHODS

The theoretical basis of this study include the business process reengineering theory, theory of software architecture, software engineering, information systems flexible theory, process-aware information system theory (Section 2.1). Research methods included literature review, case analysis, data modeling, and optimization software LINGO[3].

1. The model results from on the literature review of extending a previous theory of IS flexibility, we regard software architecture flexibility as an option that is available to the decision maker, and demonstrate deterministic treatment of software architecture flexibility tends to determine its value.

2. A comprehensive case study at civil aviation inventory control software architecture,

which depend on interviews with administered to the Travelsky [1]. Drawing upon the work of Dreyfus and Wyner (2011), as indicated in Figure 3, cost data, component and dependency data were collected through a survey Travelsky's software architecture. Through combining this data with survey, system instrumentation, and archive data, and then estimating and interpreting regression models.

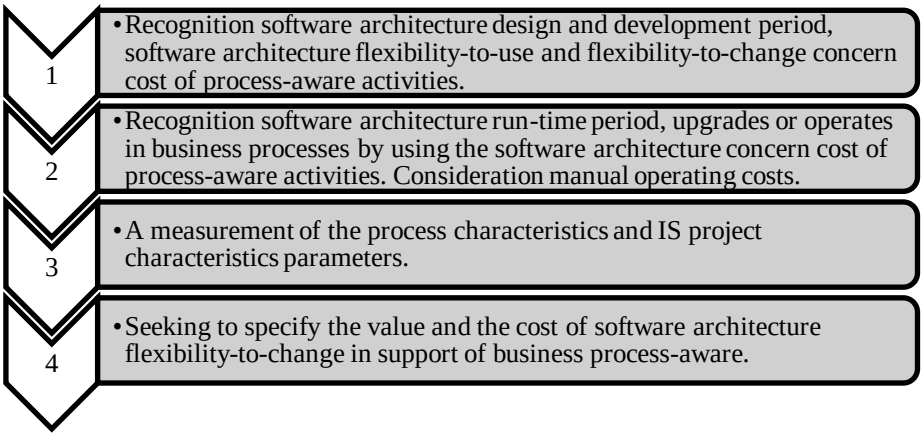

Figure 3. An evaluation Mechanism of Generation Costs of Process-aware Software Architecture

4. AN ECONOMIC MODEL TO ASSESS THE IMPACT OF SOFTWARE ARCHITECTURE FLEXIBILITY STRATEGIES ON BUSINESS PROCESS-AWARE

After researching numerous sources of scholarly writings dealing with keywords process-aware, software architecture flexibility, and software architecture cost certain common and interlinking characteristics began to emerge. Figure 4 shows the synthesized an economic model to assess the impact of software architecture flexibility strategies on business process efficiency. Ness (2005, 2011) suggested a framework that called out the relationship between "Information Technology Flexibility, IT Effectiveness, and Strategic Alignment". The center of the model comes from Ness's ITF model. In this paper, the model considers relationship between process-aware software architecture and costs of flexibility.

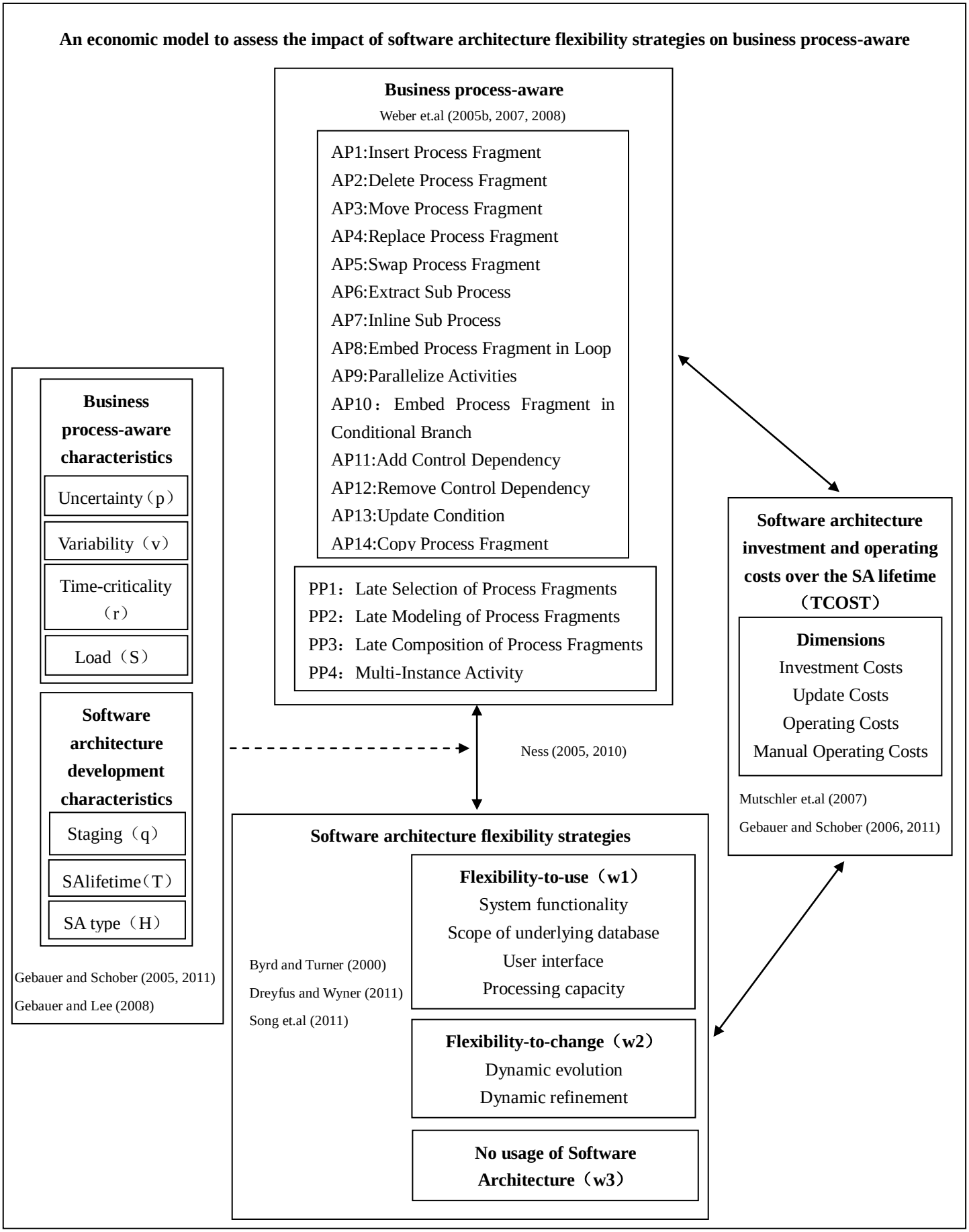

Fig4. Model Overview

4.1 Model Overview

In Fig. 4, complementing the earlier research studies, we seek to specify the value and the cost of software architecture flexibility-to-change in support of business process-aware.

Business Process-Aware Characteristics

In line with previous research about Gebauer and Schober (2006, 2011) , Gebauer and Lee (2008), we model a business process based on the following four parameters: time-criticality (r measures time-criticality), load (S means process load scaling factor with S=1 for the base scenario), variability (v measures process variability), process uncertainty (p measures process uncertainty) (Fig. 4).

Time-criticality

Aalst (2009) identifies the different phases of a process in the context of a WFMS, based on the different phases and the three mechanisms, different types of flexibility are classified in Figure 5. Time-criticality measures the share of time-critical process activities, the higher time-criticality the more outsourcing-arrangements such as commercial procurement or outsourcing contractors and partners.

	defer (decide to decide later)	*change (decide to change model)*	*deviate (decide to ignore model)*
design time	*e.g., defer to run-time by using late binding or declarative modeling*	*N/A*	*N/A*
configuration time	*e.g., defer configuration decisions*	*e.g., remodel parts of the process at configuration time*	*e.g., violate a configuration constraint*
instantiation time	*e.g., defer the selection of parameters or process fragments*	*e.g., modify model for a particular customer*	*N/A*
run time	*N/A*	*e.g., change model for running instance or migrate instance to new model*	*e.g., skip or redo a task while this is not specified*
auditing time	*N/A*	*N/A*	*N/A*

Figure 5: Classification of the different types of flexibility based on the phase and mechanism.

Dan (2010) proposes commercial procurement can be both COTS (Commercially-Off-The-Shelf) components (such as the development environment comes

with ActiveX controls, JavaBeans and Delphi components, and numerous libraries, DLL interface and API), the project can also be used contractors and partners to develop the NDI components (Non-Developmental Item).

Process load

Gebauer and Schober (2006) define a set of related activities designed to produce a specific output, such as procurement, or budget decision making can be viewed as business process. As explained by Gebauer and Lee (2008), process load is expressed by the overall activity load of the business process. We consider process load scaling factor with S=1 for the base scenario.

Variability

As explained by Gebauer and Schober (2006), distribution of different process requirements during the IS-lifetime (Lorenz-curve for graphical representation), variability refers to the degree to which process activities are concentrated on the same process task. Parameters x_1 ($0 \leq x_1 \leq 1$) relates to the number of activities and $L(x_1)$ to the number of corresponding tasks, the value of $L(x_1)$ ($0 \leq L(x_1) \leq 1$) determines the extent of flexibility-to-use that is built into the PASA from its very beginning. We use the form:

$$L(x)=x^{v}(1-(1-x)^{1-v}), \quad (1)$$

The variability parameter v measures the concentration of process activities, with $0 \leq v \leq 1$. Values of v that is close to 0 describe business processes with much the supported patterns number, thus high levels of variability, and then v is close to 1 describe business processes with little the supported patterns number, thus low levels of variability, where activities for much number of process tasks dominate. Counting the number to analyze how well a system

can deal with process change, the smaller number the higher variability.

Process Uncertainty

As explained by Gebauer and Schober (2006), uncertainty refers to the degree to which process tasks are well-understood and structured by the software architecture at the time of system initiation, the uncertainty parameter p with $0 \leq p \leq 1$ is the probability that a process task can be foreseen and described at the time of software architecture initiation. Process tasks refer to the different functionalities that a business process implies. A set of changes patterns and change support features to put forwards PASA can efficient reduce business processes uncertainty and deal with the situation of the needs business processer facing and changeable environment (Section 2.1).

Software architecture development characteristics

The characteristics of the software architecture development process are summarized by the three parameters software architecture type (H means different type of software architecture), software architecture lifetime (T means SA lifetime in years), and staging (q means percentage of the software architecture being implemented immediately) (Fig. 4).

Software architecture type

Gebauer and Schober (2006, 2011) proposed the relative importance of the different components of flexibility-to-use (through functionality, database, user interface, and processing capacity) and of flexibility-to-change (via dynamic evolution, dynamic refinement) have to be determined by factors such as the key drivers of the PASA and relative component costs. Type of software architecture are the key factor and the relative costs of the different options will be determined by the specifics of the underlying business process, but also

exhibit path dependency to the extent that previous investments in flexibility-to-change determine the availability of dynamic evolution, dynamic refinement of the PASA applicable to the current situation.

Staging

As explained by Gebauer and Lee (2008), staging reflects the fact that software architecture is often not put into operation all at once, but in consecutive stages. Software architecture management prefers a "staged" implementation approach with a certain percentage q of the software architecture being implemented immediately and the rest (1−q) being implemented subsequently during the system's lifetime. The dynamic effects of investment staging apply the annuity method, and set an average discount factor (DC) as (Schober and Gebauer, 2011):

$$DC = ((1+i)^{T}-1)/(i(1+i)^{T}T) \text{ for } i>0 \text{ and } DC=1 \text{ for } i=0, \quad (2)$$

Where the remaining investments are spread out evenly over the software architecture lifetime T in years, i denote the yearly discount rate.

Software architecture lifetime

Software architecture lifetime defines the total lifetime of the Software architecture in years. We assume software architecture lifetime generally is similar to information system lifetime.

Flexibility Strategies

The variables w_1, w_2 and w_3 express the recommended mix of flexibility strategies in response to the software architecture development and business process-aware characteristics. The weights w1, w2 and w3 are derived decision variables that are calculated by minimizing

total costs (TCOST) over the lifetime of the software architecture. However, the shares of the three different flexibility strategies can be expressed as:

$$w_1 = px_1; \quad w_2 = (1-p)x_2; \quad w_3 = p\,(1-x_1) + (1-p)\,(1-x_2) = 1-w_1-w_2, \qquad (3)$$

As explained by Schober and Gebauer (2011), the variables w_1, w_2 and w_3 are derived based on the primary decision variables x_1 and x_2 and the estimated parameter p, x_1 means share of process activities for tasks that are anticipated at the time of PASA initiation and use flexibility-to-use; x_2 means share of process activities for tasks that are not anticipated at the time of PASA initiation and use flexibility-to-change.

Software Architecture Flexibility-to-Use

Share w_1, we indicate the share of all process activities that utilize the functionality that is built into the PASA upon its inception (flexibility-to-use).

Software Architecture Flexibility-to-Change

Share w_2 refers to the activities that are handled by the PASA after it has been modified or after a new functionality has been implemented at some time during its operational lifetime (flexibility-to-change).

No usage of Software Architecture

Share w3 refers to the activities that are not processed by the PASA under consideration, but by different means, such as manually or no usage of software architecture.

4.2 Investment decisions

In the model two investment decisions have to be made before the PASA under consideration becomes operational. The first decision concerns the basic investment (ICOST) such as an investment into an “off the shelf” standard PASA and the second decision concerns

an additional investment (FCOST) such as process-aware ability.

ICOST for providing flexibility-to-use is modeled as:

$$ICOST = \{a + b\, L(x_1)\, (q + (1-q)\, DC)\}\, z, \quad (4)$$

$$z \geq 0.5(x_{1+}\, x_2), \quad (5)$$

As explained by Schober and Gebauer (2011), where a denotes base investment in flexibility-to-use at the time of software architecture initiation, and b denotes additional investment in flexibility-to-use, if all process tasks that are anticipated at the time of software architecture initiation were supported by the software architecture, if software architecture is implemented at all, a binary variable z with z=1, whereas if not z=0. To ensure meaningful results, the logical constraints need $x_1+x_2>0$, variable z has to be equal to 1. After computing endogenously by the model, the values for the decision variables x_1, x2 and z will be known.

In PASA, ICOST denotes total investment in flexibility-to-use at the time of software architecture initiation, it include such as user account information management and storage, user login authentication, access request load balancing, enterprise service bus, service management, registration services, encryption, base class library, dynamic load engine, basic workflow engine, engine logs, thread pool engine and cross-organizational information sharing-and-hidden relevant change support features (Section 2.1).

FCOST for providing flexibility-to-change is modeled as

$$FCOST=c\, y, \quad (6)$$

$$y \geq x_2, \quad (7)$$

As explained by Schober and Gebauer (2011), if flexibility-to-change is provided, the logical constraints for flexibility-to-change to be applicable (x2>0), y is a binary decision

variable with y=1, or else if not y=0. FCOST denotes actual investment in flexibility-to-change at the time of software architecture initiation. The parameter c denotes investment in flexibility-to-change if provided; it will later be endogenously calculated as the "value of software architecture flexibility".

FCOST includes only the initial investment into flexibility-to-change, a separate model term UCOST measured the actual software architecture upgrade costs using the flexibility-to-change option provided. To modify and upgrade the PASA during its lifetime, the costs UCOST are modeled as

$$UCOST = e\ L(x_2)\ DC, \quad (8)$$

As s explained by Gebauer and Schober (2006), x2 ($0 \leq x2 \leq 1$) denotes the share of process activities that are unknown at the time of PASA initiation, but that are supported by the PASA after flexibility-to-change is utilized. The parameter e denotes software architecture upgrade costs if all process tasks that are not anticipated at the time of software architecture initiation were included in the upgrade.

4.3 Ongoing operations

Software architecture operating costs (OCOST) are associated with actual software architecture operating costs, and manual costs (MCOST) are associated with actual costs for manual operations or outside of the current PASA.

OCOST of the PASA can be written as:

$$OCOST = S\ d\{(q+(1-q)L^{-1}(0.6)w_1+0.4w_2)\}T\ DC, \quad (9)$$

As explained by Schober and Gebauer (2011), the parameter d is an estimate for the software architecture operating costs if all process activities were supported by the software

architecture (i.e., w3=0). These costs are multiplied by the shares of activities w_1 and w_2 that actually utilize the PASA.

In addition, we multiply the inverse L^{-1} (0.6) with w_1, instead of 0.4 as in the case of w2. Because of mutschler (2006) proposed that PASA's additions, modifications and upgrades are generally staged 40% over the lifetime of the software architecture by a necessary redesign of business processes, 40% of the share w2 is handled outside of the system, as the corresponding activities occur before the system has been modified or upgraded.

The number of years T is multiplied with the yearly operating costs that depict the average yearly discount factor DC and the software architecture lifetime. The scaling factor S with S>0, we assume the standardized factor S=1, and evaluate different process load scenarios are out of our research.

The operating costs MCOST for activities that are performed outside of the SA are modeled as:

$$MCOST = S\ f\ (1+rg)\ \{(1-q)\ (1-L^{-1}(0.6))\ w_1+0.4w_2+w_3\}\ T\ DC, \quad (10)$$

As explained by Schober and Gebauer (2011), a yearly cost factor f multiplied the operating costs that apply in cases where all process activities are performed outside of the software architecture. Cost factor f denotes manual operating costs if all process activities were performed manually. We include a percentage cost premium g, g denotes cost markup for manually performing time-critical process activities. Cost premium g applies in cases where a share r of time-critical activities with $0 \leq r \leq 1$ is processed outside of the software architecture; we need to assume that outside processing is more expensive, and thus less time-efficient.

The model's objective function is the minimum of the total costs over the entire lifetime of the software architecture. TCOST provides the basis for the derived decision variables w_1, w_2 and w_3, and joins the primary decision variables x_1, x_2, y and z:

$$TCOST=ICOST+FCOST+OCOST+UCOST+MCOST \quad (11)$$

$$\text{Minimize } TCOST\ (0\leq x_1,x_2\leq 1, y,z\in\{0,1\}) \quad (12)$$

To solve the model constitutes a mixed-integer program, small-scale, and nonlinear, for the applications; we used the optimization software LINGO (Schober and Gebauer ,2011).

5. EVIDENCING THE EXISTENCE RELATIONSHIP BETWEEN PROCESS-AWARE SOFTWARE ARCHITECTURE AND COSTS OF SOFTWARE ARCHITECTURE FLEXIBILITY

In this section, we evidence the existence relationship between PASA and costs of software architecture flexibility, and then ready to apply our model to determine the value and the cost of PASA flexibility-to-change for a specific example. To evaluate this initial model of PASA flexibility, the views of managers in Travelsky[1] were sought and analyzed.

5.1 Case Study

Traditional aviation inventory control systems (ICS[2]) are lack of organic integration, and lack of flexibility, and they cannot change with demand, and data are scattered and should not be shared. The business process should not automatically converge. With the increasing competition in the aviation market, as well as development of civil aviation business and improvement of the level of information, PAIS for the airline’s development is becoming more and more important.

In this context on the focus of analyzing the characteristics of airlines traditional

distribution model and the challenges it has to face in the increasingly changing commerce and technology environment. Liu et.al (2009) proposed a new airlines distribution model for internet circumstance with which airlines set up an open self-controlled platform-independent hub of business and information so as to control products channels and customers and thereby asserts commercial interest through various emerging channel and technology. Liu et.al (2009) described E-commerce and call center is very important to the two direct channels. Exists between the two systems are too much from the business data sharing needs and process optimization. For example: When a customer to complete all online reservations interconnect features, hoping to upgrade the call center rescheduled, refunds refund and other operations; and if the customer call center channels function to complete all the reservations, or call center rescheduled to complete upgrades, refund, refund operation, but also hopes to interconnect online status inquiries. Therefore, the two systems want to share ticket sales, ticket service data (for upgrades, rescheduled, refund, refunds, order tracking, mailing list and other travel services, ticket process data) and share sales performance management, etc., in order to achieve between the two airlines direct sales channel sales and service processes in a unified and uniform customer experience, increase customer satisfaction and customer service levels.

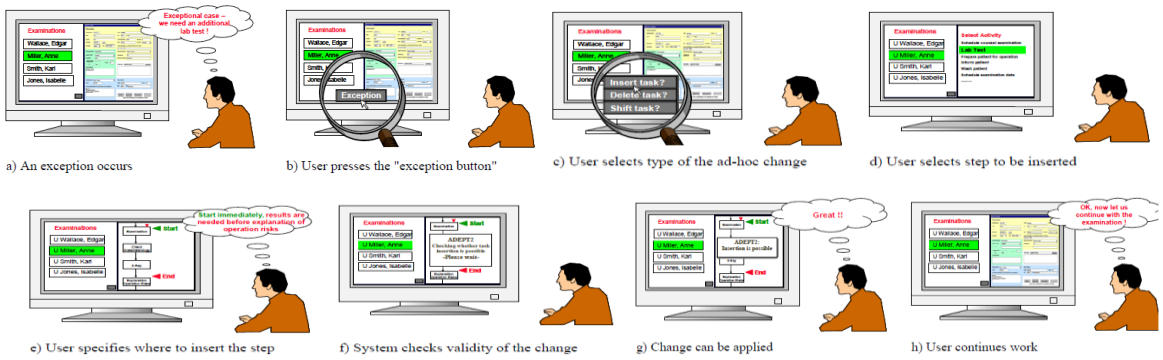

Figure 6. ICS deal with the surrounding business system requirements (adapted from Reichert et. al, 2009)

Pattern Description

To solve these problems, Travelsky use PASA to design ICS software architecture. Using process-aware thesis and technology design a common set of software framework to solve the obstacles which encountered in the information development. ICS uses SOA architecture design, two types B/S and C/S to build software architecture. ICS deal and integrated with the surrounding business process requirements are as examples (Figure 6):

(1) Usually passenger may only book a flight after it has been approved by the ICS. Exceptionally, for a particular process the booking of a flight shall be done in parallel to the approval activity; consequently the book flight activity has to be moved from its current position in the process to a position parallel to the approval activity. AP3 to deal with the application (Section 2.1).

(2) The passenger's parcels registering and scanning activity has to be repeated until all parcels are scanned in the airport counter check-in. The number of parcels is not known at build-time. PP4 to deal with the application (Section 2.1).

Support Components and Dependencies Description

A pattern-based analysis combines self-adapting and self-evolution theory in PASA shape ICS prototype system. As indicated in Figure 7, ICS include the dynamic evolution and dynamic refinement relevant support components (run-time tools).

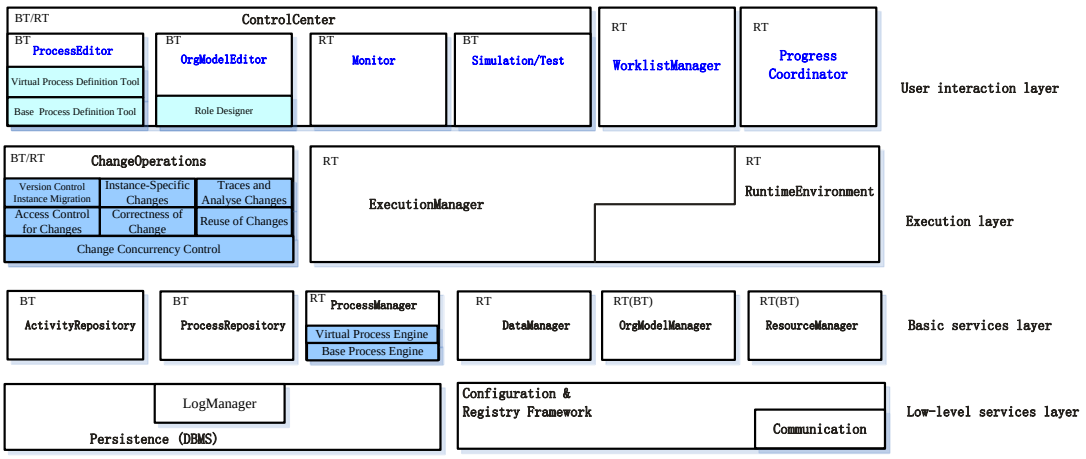

Figure 7. Process-aware software architecture of ICS support components and dependencies

As indicated in Figure 7, the Low-level Services Layer is a thin abstraction on SQL, enabling a DBMS independent implementation of persistency. The Basic Services layer is responsible for storing and locking different entities of the process management system (e.g., process templates and process instances). The Execution Layer encapsulates essential process support functions including process enactment and change management. The User Interaction layer provides different build-time (BT) and run-time (RT) tools to users, including a process editor and a process monitoring component.

As the development of run-time (RT) tools components takes more time and cost, also consider PASA in the course of the case there may need to commercial procurement. Only few of PASA provide support for both changes of individual process instances and the propagation of process type changes to a collection of related process instances. The knowledge about changes has not yet been exploited by any of these systems. To overcome this practical limitation, PASA must capture the whole process life cycle and all kinds of changes in an integrated way. They must allow users to deviate from the predefined process in exceptional situations, and assist them in retrieving and reusing knowledge about previously performed changes. Weber et.al (2009) present a proof-of concept implementation

of a learning adaptive PASA. The prototype combines the PASA for dynamic process changes with concepts and methods provided by case-based reasoning (CBR) technology (Figure11). Considerations based on time-criticality, enterprise often consider commercial procurement similar tool.

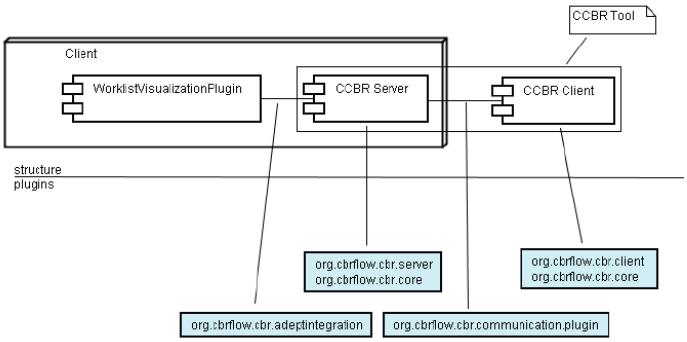

Figure 8. Structure of CCBR Tool (Weber et. al, 2009)

5.2 Value of PASA flexibility for deterministic process loads

The model parameters were collected through a survey administered to the ICS Service Owners. The model solution exhibits an interesting mix of flexibility strategies. We further assume that there exist no dependencies between these model parameters.

We measure the cost of making an architectural component change with process change pattern as explain by Dreyfus and Wyner (2011). The weights of various software architecture flexibility components can be divided based on the cost of inputs. (Table 2) Cost data, business process-aware and software architecture development characteristics parameters were collected through a survey administered to the ICS Service Owners. With component knowledge provided estimates of the parameter e cost of each architectural operation listed above for dynamic evolution and dynamic refinement components that ultimately enter the regression model as explained by Dreyfus and Wyner (2011). (Section 4.2)

To evidence the management inspiration, we use different of preferences of the business

process (variability are 0 to 1, uncertainty (p= 0.8, p= 0.5, p=0.2), no time-criticality (r= 0, r=0.5)), other related costs and parameters of data in a tabular data (Table 2). The result is that high time-criticality can strengthen the efficiency contribution of the software architecture in general; high variability can limit the efficiency of the software architecture in general and has low counting the supported patterns number; low uncertainty corresponds with software architecture flexibility-to-use and has high counting the supported patterns number; high uncertainty corresponds with software architecture flexibility-to-change.

Table 2. Mechanism of Generation ICS Costs

<table>
<tr><td colspan="4">Classification of information systems based on SAP, supported software architecture Type of different IS includes E-Commerce, CRM, SRM, SCM, PLM, PORTAL, ERP, BI, WF (Yu, 2007). ICS need to an order processing system (process-oriented) software architecture to support its operation, such as WF, the two components of processing capacity and variability of access methods (user interface) may be the main drivers of flexibility-to-use, and the two components of dynamic evolution and dynamic refinement may be the main drivers of flexibility-to-change or consider commercial procurement similar tool (Section 5.1).</td></tr>
<tr><td colspan="4">Cost-related parameters: a=125.67, b=367.58, c=63.75 (later c=0), d=175.45, e=324.56, f=467.24 (Unit: Ten Thousand CNY); g=0.5; i=0.05.</td></tr>
<tr><td colspan="4">Parameters that characterize the business process: p=0.8, v=0.6, r=0.1 and S=1.
Parameters depicting the software architecture development and implementation project and software architecture lifetime: q=0.5 and T=3.</td></tr>
<tr><td>Software architecture flexibility to use (w_1)</td><td>Software architecture flexibility to change (w_2)</td><td rowspan="3">No usage of software architecture (w_3)</td><td rowspan="3">Total costs over the entire lifetime of the software architecture (TCOST)</td></tr>
<tr><td>System functionality (weight:0.24)</td><td rowspan="2">If y=1,
Dynamic evolution
(weight:0.62)</td></tr>
<tr><td>Scope of underlying database (weight:0.12)</td></tr>
</table>

User interface (weight:0.36)	Dynamic refinement (weight:0.38)		
Processing capacity (weight:0.42)			
0.72	*0.07 (y=1, x1=0.9,x2=0.34)*	*0.21*	(*TCOST*) 1113.16
0.72	*0 (c=0,x_2=0,y=0,x1=0.9)*	*0.28*	(*TCNF*) 1067.31
0.72	*0.07 (c=0,x1=0.9,x2=0.34)*	*0.21*	(*TCF*) 1049.41

c* (Value of software architecture flexibility-to-change) = TCNF−TCF =1067.31-1049.41 = 17.9

The cost difference is 1113.16−1067.31=45.85 monetary units. In other words, had we been able to reduce the investment into flexibility-to-change by that amount (c*=63.75−45.85=17.9), the investment would have been included in the optimal solution.

c'(Software architecture flexibility-to-change) = TCOST * w_2= 1113.16 * 0.07 = 77.92

Following the systematic analysis of the model, we have turned out that the two components of processing capacity (42%) and user interface (36%) may be the main drivers of flexibility-to-use; the two components of dynamic evolution (62%) and dynamic refinement (38%) may be the main drivers of flexibility-to-change; process-aware software architecture costs have to be justified by benefits, because of c*>0.

Note: TCNF (total costs with no software architecture flexibility-to-change provided); TCF (total costs with software architecture flexibility-to-change provided);

6. CONCLUSIONS AND FUTURE RESEARCH AGENDA

The emergence of different process support paradigms and the lack of methods for comparing existing change approaches have made it difficult for PASA engineers to choose the adequate technology. This makes it difficult for PASA engineers to assess the maturity and the change capabilities of those technologies, often resulting in wrong decisions and expensive investments. Therefore, we need to provide a detailed research on the relationship between software architecture and costs of flexibility.

Motivated by the relationship observation, we extend a previous theory of software architecture flexibility, and consider software architecture flexibility, and evaluate weights of various software architecture flexibility components, and concrete consideration of dynamic

aspects, including software architecture life time. Accordingly, we propose an economic model to assess the impact of software architecture flexibility strategies on business process-aware. Based on the model we have done related research and obtain the corresponding results:

1. To evidence the management inspiration is that high uncertainty corresponds with software architecture flexibility-to-change and has high counting the supported patterns number; low uncertainty corresponds with software architecture flexibility-to-use; high variability can limit the efficiency of the software architecture in general and has low counting the supported patterns number. High time-criticality can strengthen the efficiency contribution of the software architecture.

2. A comprehensive case study at ICS, we describe the prototype of process-aware software architecture, and consequently will support PASA engineers in selecting the right technology for realizing flexible PASA. Then we measure specifies the value and the cost of process-aware software architecture flexibility-to-change, and process-aware software architecture costs have to be justified by benefits. In addition, the weights of various software architecture flexibility components can be divided based on the cost of inputs.

Our futures works include value of PASA flexibility for stochastic process loads approach; real option analysis (ROA) treatment of PASA flexibility tends to estimate its value in particular in low-risk; valuate additional academic and commercial systems with internal-organizational and cross-organizational process-aware software architecture. In additional, we also consider the model to support the different types of software architecture.

7. REFERENCES

van der Aalst WMP, Basten T. (2002). Inheritance of workflows: An approach to tackling problems related to change. Theoretical Computer Science,270(11):125−203. [doi: 10.1016/S0304-3975(00)00321-2] .

W.M.P. vander Aalst, M.Weske. (2005). Case handling : a new paradigm for business process support, Data&Knowledge Engineering53(2) 129–162.

W.M.P. van der Aalst, et al. (2007). Business process management: where business processes and web services meet, Data & Knowledge Engineering 61(1) 1–5.

Wil M.P. van der Aalst. (2009). Process-Aware Information Systems: Design, Enactment, and Analysis Published Online: 16 MAR 2009 DOI: 10.1002/9780470050118.ecse577.

M.Adams, A.H.M.terHofstede, M.LaRosa. (2011). Open source software for workflow management: the case of YAWL,IEEE Software28(3) 16–19.

Anoint P A. (2003). Distributed Adaptable Software Architecture Derived From Component Model [J].Computer Standards & Interfaces, Special issue: Adaptable Software Architectures.

J.vom Brocke, T.Sinnl. (2011). Culture in business process management: a literature review, Business Process Management Journal17(2) 357–378.

Byrd, T. A. and D. E. Turner. (2000). “Measuring the Flexibility of Information Technology Infrastructure: Exploratory Analysis of a Construct,” Journal of Management Information Systems, 17 (1), pp. 167-208.

Ruey-Shun Chen et.al. (2011). Factors Influencing Information System Flexibility An Interpretive Flexibility Perspective China University of Technology, Taiwan.

P.Dadam, et al. (2009). Von ADEPT zur Arista Flow BPM Suite—Eine Vision wird Realit¨at: “Correctness by Construction” undflexible, robust Ausf ¨ uhrung von Unternehmensprozessen,EMISAForum29(1) 9–28.

M. Dumas, W.M.P. van der Aalst, A.H.M. ter Hofstede (Eds.). (2005). Process Aware Information Systems: Bridging People and Software Through Process Technology, John Wiley & Sons, Hoboken, New Jersey.

Duncan, N. B. (1995). Capturing flexibility of information technology infrastructure: A study of resource. Journal of Management Information Systems, 12(2), 37–57.

David Dreyfus, George Wyner. (2011) . AIS Electronic Library (AISeL) Digital Cement Software Portfolio Architecture, Complexity, and Flexibility.

T.Erl. (2005). Service-Oriented Architecture: Concepts, Technology, and Design, Prentice-Hall, Upple Saddle Revier, New Jersey.

J. Gebauer, F. Schober. (2006). Information system flexibility and the cost efficiency of business processes, Journal of the Association for Information Systems 7 (3).

Judith Gebauer, Fei Lee.(2008). Enterprise System Flexibility and Implementation Strategies−Aligning Theory with Evidence from a Case Study College of Business, Information Systems Management, 25: 71–82.

M. Hammer, S. Stanton. (1995). The Reengineering Revolution - The Handbook, Harper Collins Publ.

LI Xiao, MA Xiao-jun, Research on Software Structure Based on Self- adapting, Software Designing and Development, 2008

LI Jing-jie, WANG Wei-ping, YANG Feng. (2010). Review on approaches of flexible workflow, Computer Integrated Manufacturing Systmes, Vol.16 No.8.

LIU Wei, ZHU Xiao-hong. (2009). E-commerce core platform for airlines base on SOA Journal of Computer Applications Vol. 29 Dec. 2009.

Bela Mutschler, Johannes Bumiller.(2006). Why Process-Orientation is Scarce: An Empirical Study of Process-oriented Information Systems in the Automotive Industry, EDOC '06 Proceedings of the 10th IEEE International Enterprise Distributed Object Computing Conference, IEEE Computer Society Washington, DC, USA, 2006.

Mutschler, Manfred Reichert, and Stefanie Rinderle.(2007). Analyzing the Dynamic Cost Factors of Process-Aware Information Systems: A Model-Based Approach, Springer-Verlag Berlin Heidelberg 2007Bela J. Krogstie, A.L. Opdahl, and G. Sindre (Eds.): CAiSE 2007, LNCS 4495, pp. 589–603.

Ness, L. R. (2005). "Assessing the Relationships Among Information Technology Flexibility, Strategic Alignment, and Information Technology Effectiveness," Ph.D. dissertation, 2005.

NESS. (2010). IT FLEXIBILITY: A SYNTHESIZED MODEL FROM EXISTING LITERATURE ISSN #1042-1319 TOM FULLERTON DR. LAWRENCE R. Journal of Information Technology Management Volume XXI, Number 3.

J.Recker. (2010). Continued use of process modeling grammars: the impact of individual difference factors, European Journal of Information Systems 19(1) 76–92.

J.Recker, et.al. (2011). Doon to logical deficiencies in modeling grammars matter? MIS Quarterly 35(1) 57–79.

Reichert M, Rinderle S, Dadam P. (2003). On the common support of workflow type and instance changes under correctness constraints. In: Meersman R, et al., eds. Proc. of the OTM Confederated Int'l Conf. LNCS 2888, Berlin, Heidelberg: Springer-Verlag, 407−425. [doi: 10.1007/978-3-540-39964-3_26].

Reichert M, Dadam P. (2009). Enabling adaptive process-aware information systems with ADEPT2. In: Handbook of Research on Business Process Modeling. Hershey: Information Science Reference, 2009. 173−203.

H.A.Reijers, S.Limam, W.M.P. vander Aalst. (2003). Product-based work-flow design, Journal of Management Information Systems20(1) 229–262.

H.A.Reijers, S.L.Mansar. (2005). Best practices in business process rede- sign: an overview and qualitative valuation of successful redesign heuristics, Omega33(4) 283–306.

Franz Schober, Judith Gebauer. (2011). How much to spend on flexibility? Determining the value of information system flexibility, Decision Support Systems.

Soh, C., S. K. Sia, and J. Tay-Yap. (2000). "Cultural Fits and Misfits: Is ERP a Universal Solution?" Communications of the ACM, 43 (4), pp. 47-51.

Weber, B., Reichert, M., Rinderle, S., & Wild, W. (2005b). Towards a framework for the agile mining of business processes. In Proceedings of the BPM'05 Workshops, Nancy, France, LNCS 3812, pp. 191- 202.

B. Weber, S. Rinderle, M. Reichert. (2007). Change Support in Process-Aware Information Systems – A Pattern-Based Analysis, Tech. Rep. TR-CTIT-07-76, CTIT, University of Twente, The Netherlands.

Weber B, Reichert M, Rinderle-Ma S. (2008). Change patterns and change support features—Enhancing flexibility in process-aware information systems. Data & Knowledge Engineering, 66(3):438−466. [doi: 10.1016/j.datak.2008.05.001].

B. Weber, M. Reichert, W. Wild, S. Rinderle-Ma. (2009). Providing integrated life cycle support in process-aware information systems. World Scientific Publishing Company, Int'l Journal of Cooperative Information Systems (IJCIS) , Vol. 18, No. 1, pp. 115-165.

B.Weber, B.Mutschler, M.Reichert. (2010). Investigating the effect of using BPM technology: results from a controlled experiment, Science of Computer Programming75(5) 292–310.

Wen Yu. (2007). speculative concept of software architecture, programmer.

Song W, Ma XX, Hu H, L ü J. (2011).Dynamic evolution of processes in process-aware information systems. Journal of Software, 22(3):417−438.

Xuyi Dan. (2010). Software reuse in software architecture research, Institute of Information Science, Silicon Valley, 2010.7.

Zhang Geng. (2009). Information technology-based strategic analysis of business processes flexible ： Technology Management Research," No. 10.

Notes

---

1 Travelsky

Travelsky is formerly the Civil Aviation Information Center. Peng (2005) described Travelsky bear the civil reservation, ticket sales, airport check-in, loading, freight and other business systems development, operation and protection work, after nearly 20 years of development, Travelsky information services is very broad, covering all areas and the

majority of China's civil aviation flight passengers.

2 ICS

Travelsky have finished in the construction of base on a legacy systems to a new generation of core system, the new flight data control system to achieve a full automation process can be automated from the flight planning system to collect flight data, flight data packets generated, sent the letter in the ICS. ICS full name is the Inventory Control System, which airline to use the airline reservation system. ICS is centralized, multi-carrier system. Each airline have their own separate database, independent of the user base, independent of the control and management of various operations are to be personalized, including flight schedules, seat controls, tariffs and revenue management, airline alliances, sales control parameters and other information and a complete set of reservation functionality engine.

ICS system updates the entire flight season about 300,000 flights only need more than 1 hour, you can automatically 500 classes to 600 classes' daily flights to adjust, change the manual entry of artificial control, significant savings in labor costs. The new system uses PASA, simple, convenient query data is accurate, and for the spring 2012 China Southern Airlines flight information release provides a powerful support.

3 LINGO

LINGO is a simple tool for utilizing the power of linear and nonlinear optimization to formulate large problems concisely, solve them, and analyze the solution. Optimization helps you find the answer that yields the best result; attains the highest profit, output, or happiness; or the one that achieves the lowest cost, waste, or discomfort. Often these problems involve making the most efficient use of your resources including money, time, machinery, staff,

inventory, and more. Optimization problems are often classified as linear or nonlinear, depending on whether the relationships in the problem are linear with respect to the variables.