International Journal of Electrical, Electronic and Communication Sciences
ISSN: 2517-9438
Vol:5, No:11, 2011

TSM: A Design Pattern to Make Ad-hoc BPMs Easy and Inexpensive in Workflow-aware MISs

Haitao Yang

***Abstract*—**Despite so many years' development, the mainstream of workflow solutions from IT industries has not made ad-hoc workflow-support easy or inexpensive in MIS. Moreover, most of academic approaches tend to make their resulted BPM (Business Process Management) more complex and clumsy since they used to necessitate modeling workflow. To cope well with various ad-hoc or casual requirements on workflows while still keeping things simple and inexpensive, the author puts forth first the TSM design pattern that can provide a flexible workflow control while minimizing demand of predefinitions and modeling workflow, which introduces a generic approach for building BPM in workflow-aware MISs (Management Information Systems) with low development and running expenses.

***Keywords*—**Ad-hoc workflow, BPM, Design pattern, TSM

I. Introduction

It is a controversial issue to discuss the merit and demerit of the many years' academic research and application practice of BPM (Business Process Management). Many practitioners, as we know doubt are so many efforts really useful and cost-effective for building a workflow-aware MIS? It seems that most of BPM vendors, developers, and academic researchers, to a great degree, tend to make things more complex and expensive than need to be in this workflow-aware issue. Although nowadays it is a usual practice for many IT consulters to suggest a product solution of standalone BPMs, however, in many cases a simple methodological approach, from our point of view, is good enough to implement MISs capable of dealing ad-hoc workflows well; there is no need for those so-called intensive product or work. Under current approaches, modeling process or workflow is an inevitable step [1], which usually makes BPM more complex and costly than it needs to be for many situations. Apart from disposing regular workflow, we sometimes need a flexible WfMS (Workflow Management System) to support ad-hoc workflow. An ad-hoc workflow or casual workflow, in short Ah-FL, refers to less-confined or less-regular workflows, which are often featured with necessary human intervention that might affect the practical route of workflow at any node and any time in the course, e.g., they can be encountered in developing a MIS for Adhocracies organizations [2]. Ah-FL is not negligible simply because

1) Anomalistic workflows exist in reality though they are less frequent and desired;
2) Workflow exceptions exist due to situation change or system incapability, etc.We must point out that anomalistic flows and flow exceptions are often confused [3], [4]. In a well taxonomy, exception handling belongs to the subject of reliable design; whereas anomalistic flows refer to those that behave in an unusual way. In workflow-aware MIS applications, process automation is typically not a strict requirement for an Ah-FL, and an abnormal routine might start at any node of flow process at any time. There is a so-called 80/20 ratio phenomenon of the regularized to ad-hoc cases [5], i.e., roughly speaking an 80 percents or so of the workflow cases belong to the regularized category, about 20 percents are ad-hoc (casual). It is different from that of production systems in the manufactory industries [6].

Patterns of anomalistic flows can be classified into two categories: the expected and the unexpected, "the former refers to those that are known in advance to the workflow designer, whereas the disposal of the later typically resorts to a human intervention" as stated in [7]. Even for the expected anomalistic workflows, when and which of them should occur might be unpredictable despite their patterns could be enumerated in advance [4], that is, their occurrences might be predictable or unpredictable. All these contexts of reality create need for WfMSs capable of supporting ad-hoc workflows.

II. The Problems Of Current Bpm Approaches

Here we focus on implementing a software function to cope flexibly with anomalistic flows. There are several major obstacles for classical approaches to design such a flexible WfMS:

(1) High complexity and overhead are introduced inevitably by specifying and disposing expected exceptions [4] to cope with changing application semantics, which counteracts the automation and regulation merits of WfMSs.

(2) In many business, some of their expected anomalistic flows could only be effectively described in a natural language, thus it is impossible for a software system being artificiality of the contemporary era to understand such an abnormity.

(3) Hard to cover unexpected abnormity of workflows in a coherent way [8], [9].

(4) Solutions from industries are usually product-targeted. Commercial workflow products are expensive, open source solutions needs much improvement [10].

(5) Approaches from academic societies often suffer a heavy toll of implementation or not at the stage of practicality.

(6) Lacking interoperability [11].

To overcome these problems we propose a generic approach of BPM design pattern basing on the TSM (transceiver-similar

H. Yang is with the Guangdong Provincial Construction Information Center, Guangzhou, China (phone: +86-133 4288 3896; fax: +8620-8725 1025; e-mail: yanght@gdcic.net, Haitao_yang@189.cn).

mechanism) for ad-hoc BPMs.

III. The Proposed TSM Approach

Instead of basing on workflow definition, an alternative approach is naturally outlined when each node involved in a workflow hands over tasks to their next nodes according to their next stop's address, and takes over tasks by filling in the records of sending&receiving log. The core of TSM is an address book-based mechanism of dispatching and receiving tasks, which links sequential activities into a workflow sequence. The idea of TSM pattern has the following kernel content: 1) transitions of an ongoing workflow are ignited by the manipulation of sending a task to designated addresses by participants in the current node of the workflow; 2) the addresses to which participants can send task objects are confined by the current accessible lists of address; 3) the maintenance of the relational database table of Sending Record (SR) embodies the implementation of *dispatch* and *receive* operations, which is carried out centrally by the TSM service, where the SRs table functions as a bridge between activities in a process instance. TSM take on two basic tasks: 1) address configuration management, 2) maintaining SRs to implement a dispatching and receiving mechanism.

III.i. TSM Address Book Configuration

In TSM a workflow is realized via a sequence of activities being interlinked by *dispatch* or *receive* operations, so TSM is referred to as a transceiver-similar workflow mechanism, where the Address Book (AB) configuration is the core of system management. Generally AB configuration has two basic ways: central and distributed. We assume the Centrally Configuring AB (CCAB) way, regarding that enforcing workflow per se reflects an essence of demanding stronger control.

In CCAB, there is a Basic AB (BAB) that flatly lists all addresses of all participants. Apart from BAB, there could be many Classified ABs (CABs) derived from the BAB or even other CABs. Each CAB is applied to a specific type of business process; the items of a CAB can be subcategorized into different address groups (AG). We have,

<**CAB** item> ::= <**CAB**> | <**TAI**>, <**CAB**> $\subseteq$ <**BAB**>,

<**AG**> ::= {<**TAI**>} $\subseteq$ a specific <**CAB**>

Where, TAI (Terminal Address Item) is an item indicating a specific primitive address without sub-item. The related basic data structures include:

TAI: (Address, Subject, Post, <Additional fields>)

S1: **AB** collection (Book_ID, **TAI**)

S2: Parent-Child association (Book_ID, Parent_ID)

III.ii. Operation of TSM

During a workflow process of dispatching and receiving a flowing task (FT), the FT is not moved actually, which resides where it was created in the database. An action of passing an FT is indicated by a corresponding SR. The operation of TSM is outlined as follows:

(1) Each FT has its set of SRs, named sSet in short. At the beginning, each sSet is empty.

(2) Each sending operation will trigger the TSM engine to create an sSet, which inserts one unique SR in the corresponding sSet for each addressee of the sending action. In each SR there is a unique addressee that is defined by a target address. The creation of an SR indicates that the sender has started to hand over the designated FT to the addressee defined in the SR for a succedent disposal.

(3) The reception of an FT can be handled manually or automatically, which is up to the practical design of the related application, not the focus of TSM.

(4) The finish of an FT's handover to a specific addressee is indicated by the receiver's sign-for, before which the addresser can withdraw the handover just by deleting the corresponding SR. Nevertheless an SR bearing a valid signature of reception can not be deleted.

(5) An FT could be sent to multiple addressees; in this case the sending action will create multiple SRs with one SR for one addressee distinctly.

(6) The finish of an FT's handover indicates that the disposal of the FT in the addresser's hand for that specific round of workflow is complete.

(7) Each time the same FT passing the same node in a path of workflow, its corresponding SR should have a distinct SR identity.

The data structure of SR is defined by relation S3:

<*FT_ID, SN, p-SN, S-Addr, R-Addr, sign-for, sign-time, send-time, transfer-time*>.

Where, field <S-Addr> stands for the addresser's address, <R-Addr> for the addressee's address, the values of both <S-Addr> and <R-Addr> should come from the address field of TAI in an AB (see section III.i); <p-SN> stands for the <SN> of the predecessor SR, of which the addressee is the addresser of the current SR; <sign-for> is to be filled in with a valid signature of the addressee (or whose attorney) , <sign-time> is for noting down the time when the <signs-for> is filled in, <send-time> for recording the time when the SR is created, and <transfer-time> for the time when the addressee has forwarded the FT to the next node in the workflow.

The TSM-based design pattern for workflow software is summarized as in Fig. 1.

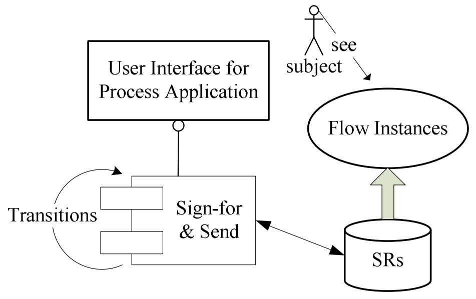

Fig. 1 TSM operation pattern illustration

International Journal of Electrical, Electronic and Communication Sciences
ISSN: 2517-9438
Vol:5, No:11, 2011

III.iii. Employing TSM in a MIS

TSM should be employed at the design stage of developing a BPM function for a workflow-aware MIS. It is the duty of workflow client applications to categorize the received FT by information from semantic attributes in the SR, and to interface the users with their FTs in a proper way. A node interface design should link each local accessible AB (derived or original) to a proper category via which the local FT outlet is controlled. Participants can pick the next addressees from their accessible ABs to transfer their current FTs, which is up to the working rules they should know. Participant's any breach to the post regulation can be easily audited from the sSet without exception. Besides, any mistakenly sent FTs could be returned to the sender via a normal sending action in the reverse direction.

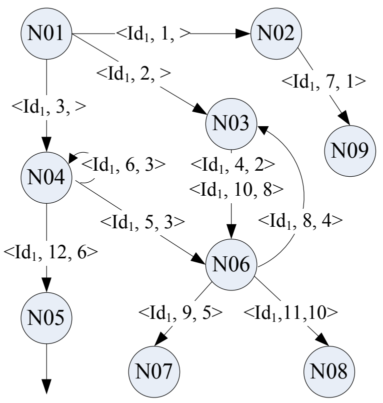

Fig. 2 Trace Diagram of Flow

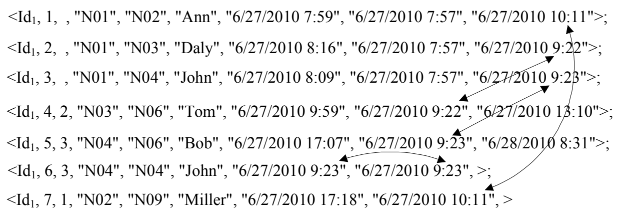
<Id$_1$, 1, , "N01", "N02", "Ann", "6/27/2010 7:59", "6/27/2010 7:57", "6/27/2010 10:11">;
<Id$_1$, 2, , "N01", "N03", "Daly", "6/27/2010 8:16", "6/27/2010 7:57", "6/27/2010 9:22">;
<Id$_1$, 3, , "N01", "N04", "John", "6/27/2010 8:09", "6/27/2010 7:57", "6/27/2010 9:23">;
<Id$_1$, 4, 2, "N03", "N06", "Tom", "6/27/2010 9:59", "6/27/2010 9:22", "6/27/2010 13:10">;
<Id$_1$, 5, 3, "N04", "N06", "Bob", "6/27/2010 17:07", "6/27/2010 9:23", "6/28/2010 8:31">;
<Id$_1$, 6, 3, "N04", "N04", "John", "6/27/2010 9:23", "6/27/2010 9:23", >;
<Id$_1$, 7, 1, "N02", "N09", "Miller", "6/27/2010 17:18", "6/27/2010 10:11", >

Fig. 3 An Instance of sSet

III.iv. Illustration of TSM's Flexibility and Expressiveness

TSM can accommodate any of FT flow traces, and can handle well any transfer at any moment during a workflow process, as is exemplified in Fig. 2, in contrast against the current WfMS products on the expressiveness of arbitrary course [12]. To be simple, in Fig. 2 we extract a tuple of three attributes <FT_ID, SN, p-SN> from the SR relation for just illustration. Fig. 2 demonstrates a flow scenario of an FT with <FT_ID> = "Id$_1$", where the FT was forked at nodes {N01, N04, N06}, and circumfluence occurred at node N06. The workflow in Fig. 2 contained a series of traces:

1: <Id$_1$,1,>→<Id$_1$,7,1>
2: <Id$_1$,2,>→<Id$_1$,4,2>→<Id$_1$,8,4>→<Id$_1$,10,8>→<Id$_1$,11,10>
3: <Id$_1$,3,>→<Id$_1$,5,3>→<Id$_1$,9,5>
4: <Id$_1$,3,>→<Id$_1$,6,3>→<Id$_1$,12,6>...

For example, after node N09 received the FT of Id$_1$ from node N02, and before node N06 sent back the FT of Id$_1$ to node N03, the snapshot of the sSet is exemplified as in Fig. 3, where, to be terse and illustrative, we just identify each address with the corresponding node's name, and using an arrow-headed line to link an SR's <send-time> to its predecessor's <transfer-time>. To present those forking branches which are not simultaneous, as branches <Id$_1$,5,3> and <Id$_1$,12,6> emitted from node N04 in Fig. 3, we have to insert a circumfluence, e.g., <Id$_1$,12,6>, at the forking node. In the case of circumfluence SR, its <sign-time> is equal to the <signs-for>, as seen Fig. 3.

As the above example shows, the trace diagram of flow can intuitively presents all forking traces of a workflow, and further all temporal sequences of transitions of workflows can be sufficiently described and logged by TSM just in one set of SRs no matter how complex the flows will be.

TABLE I
COMPARISON BETWEEN TSM AND CURRENT APPROACHES

Factors	TSM's	Commercial	Academic
Expense/workload	inexpensive	expensive	Heavy workload
Complexity	low	variable in cases	usually high
flexibility	methodological	Integration-oriented	definition-oriented
automation	low	high	variable
maintenance	easy	hard	variable, often hard
interoperability	inherent	extrinsic	extrinsic, or limited
Influence on MIS	easy integration	dependant on vendors	turn to be heavy

IV. COMPARING TSM WITH CURRENT BPM APPROACHES

TSM is a design pattern, which represents a methodological solution, applicable to designing any Ah-FL involved system, and TSM's main modules can be encapsulated and embedded in various applications. Of course, instances of TSM could also run as a standalone workflow engine. An obvious advantage of TSM is its simple implementation, according to the experience of our developing team, for example, a skillful programmer oneself can code all the basic codes of a TSM implementation from scratch within one week, there comes the cost advantage of our TSM approach too. As to the cost issue of BPM commercial product, according to [13], "the investments to purchase BPM software is hefty, in addition, BPM cost includes training, maintenance contracts, customization and development of applications, support and administration costs and finally, implementation expenses." And according to [14], "for a typical implementation that leverages a leading BPMS, the budget will be for $250,000 to $500,000 to address a meaningful process in employed organization", which contrasts with the cost of our TSM approach that only costs a week wage of common skillful programmer. The comparison of TSM's solution with current approaches' is in brevity listed in Table I.

V. CONCLUSION

MIS project practices from our software development team have validated that the implementation of TSM is so simple such that its example is no more than a pure document of programming, contrasting with today commonly complicated and often expensive practices in IT industries. Regarding that an integration of heterogeneous systems usually comes with the cost of higher complexity, lower running efficiency, and harder maintenance, a concise design pattern solution like TSM shall serve better for the same purpose of supporting Ah-FLs than those of product integration level. Empirically, it is the TSM design pattern that should be suggested instead of workflow products for dealing with Ah-FLs at ease under a low expense. Furthermore, we can explore a combination usage of TSM's and classical approaches, especially for BPMs of strictly-regular workflows where both automation and flexibility are weighted.

Acknowledgment

The Author thanks all members of his software development team for coding the TSM design pattern in related projects.

References

[1] W. M. P. van der Aalst, and A. H. M. ter Hofstede, "Workflow patterns initiative," http://www.workflowpatterns.com/
[2] S. Carlsen, J. Krogstie, and O. I. Lindland, "Evaluating flexible workflow systems," in *Proc. 30th Hawaii Int'l Conf. on System Sciences*, 1997, vol.2, pp. 230–239. DOI: 10.1109/HICSS.1997.665502
[3] C. Hagen, and G. Alonso, "Exception handling in workflow management systems," *IEEE Trans. Software Eng.*, vol. 26, no. 10, pp. 943–958.
[4] F. Casati, S. Ceri, S. Paraboschi, and G. Pozzi, "Specification and implementation of exceptions in workflow management systems," *ACM Trans. Database Syst.*, vol. 24, no. 3, pp. 405–451.
[5] Staffware Company, "Workflow patterns according to Staffware," http://www.workflowpatterns.com/vendors/documentation/vc_staffware.pdf
[6] R. Bastos, and D. Ruiz, "Towards an approach to model business processes using workflow modeling techniques in production systems," in *Proc. 34th Annual Hawaii Int'l Conf. on System Sciences*, CA: IEEE Press, 2001, vol. 9, pp. 9036–9045. DOI: 10.1109/HICSS.2001.927231
[7] W. Barbara, S. Shazia, and R. Manfred, "Beyond rigidity-dynamic process lifecycle support: a survey on dynamic changes in process-aware information systems," *Computer Science – Research and Development*, 2009, vol. 23, no. 2, pp. 47–65.
[8] M. Adams, D. Edmond, and A. H. M. ter Hofstede, "The application of activity theory to dynamic workflow adaptation issues," in *proc. 7th Pacific Asia Conf. on Inf. Syst.*, Adelaide, South Australia, 2003, pp. 1836–1852.
[9] A. Borgida, and T. Murata, "Tolerating exceptions in workflows: a unified framework for data and processes," in *Int'l Joint Conf. on Work Activities, Coordination and Collaboration* (WACC'99), pp. 59–68. San Francisco, CA: ACM Press, 1999.
[10] P. Wohed, A. H. M. ter Hofstede, N. Russell, B. Andersson, and W. M. P. van der Aalst, "On the maturity of open source BPM systems," *BPTrends*, 2009, vol. 7, no. 6, pp. 1-11.
[11] J. Xinhua, and Z. Lina, "Inter-operation of distributed workflow engine on asynchronous web services," in *proc. 3rd Int'l Conf. on Semantics, Knowledge and Grid*, 2007, pp. 590-591.
[12] Z. Marco, W. M. P. van der Aalst, R. Nick, L. Philipp, and W. Hannes, "An analysis of windows workflow's control-flow expressiveness," in *proc. 7th IEEE European Conf. on Web Services*, 2009, pp. 200-209.
[13] N. Y. Chow, "How to calculate ROI for your BPM Project", 2010. http://www.bpmenterprise.com/content/c060529a.asp
[14] C. Mark, and P. Paul, "Business process management (BPM) Definition and Solutions," 2007, http://www.cio.com/article/106609/Business_Process_Management_BPM_Definition_and_Solutions?page=5

Haitao Yang. This author received his Eng. Bachelor in mechanics, MS degree in computational mathematics, and PhD degree in computer software and theory, from the National University of Defense Technology, Sun Yat-Sen University, and the Institute of Computing Technology of Chinese Academy of Sciences, China, respectively in 1983, 1989 and 2008. He has been engaging in software industry over two decades, with a rich experience of programmer, designer, analyst, researcher, and project manager. In recent years, his major research interest is on Internet computing and information system engineering.