RC24997 (C1005-018) May 17, 2010
Computer Science

IBM Research Report

Model-Based Dependency Analysis in Service Delivery Process Management

Feng Liu, Qian Ma, Krishna Ratakonda*, Hao Wang, Liang Liu, Ying Chen
IBM Research Division
China Research Laboratory
Building 19, Zhouguancun Software Park
8 Dongbeiwang West Road, Haidian District
Beijing, 100193
P.R.China

*IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

Model-Based Dependency Analysis in Service Delivery Process Management

Feng Liu, Qian Ma, Krishna Ratakonda, Hao Wang, Liang Liu, Ying Chen

Abstract—**Build up well-defined and optimized service process is the key to deliver good service quality and service satisfaction. An effective method of dependency analysis in the service delivery process is the core to construct an optimized process. However, with increasingly complicated services, there is the large amount of the tasks elements with extreme complex dependency in the service process. Under this situation, set up correct relationship among tasks becomes time consuming and error prone. In this paper, we propose a model-based dependency analysis to automatically build up the dependency relationship among tasks in the service delivery process. It addresses the problem in analyzing dependency firstly and our approach is given. Based on the dependency analysis, we also propose several advanced analysis features on the process to guide user to optimize the process via reducing the process cost. Also a tool adopting our approach has been implemented and introduced. Based on the tool, a case study about the test service delivery process is illustrated to show the results.**

I. INTRODUCTION

Increasingly, with the severe competition for customers, the service quality and the service satisfaction has been paid more and more attentions[1]. One effective way to improve the customer satisfaction is that provide good service quality. If a service is delivered following a well-defined and optimized process, it must be helpful to improve the service quality and service satisfaction for the customer. Now the key issue is that how to build up the service delivery process in a process management system.

Process management [2] is an important concept on many domains not only in software development but also in the service delivery. A well-defined and optimized process can help customer to reduce the cost and enhance the productivity. And eventually, it can definitely improve the quality of the product as well as accelerating deliver to the market. Generally, the basic element in the process is the task which performs the detail activity step by step defined by the process. A core problem in how to construct the tasks and build up to a well-defined process is to analyze the dependency relationship. In a common simple process, most relationships are straightforward and can be set up manually. Even in a little more complex process it also can be accepted by leveraging the knowledge from domain expert. But with increasingly complex service process and dramatically increased of the number of task scale, the relationship in the process becomes ever more complicated. The manual way to configure the task dependency relationship becomes unacceptable with time consuming and error prone. The existing methods to capture the dependency are focusing on using fault injection which is usually used in a distributed system [3][4]. These methods can discovery the dependency and aid in problem diagnosis among the different component in distributed application environment. It can solve the problem in distribute system environment but can't suite for the service delivery process management. Another existing approach is scenario-driven to trace dependency analysis proposed in [5]. It only focuses on software development process which based on the software model and software engineering. Also it hardly suit for a more general service delivery process.

In this paper, we propose an approach to automatically analyze the dependency relationship in the service delivery process which is based on process model. First of all, we would propose a process model to describe the process and related element within the process. After the model definition the approach overview and algorithms will be given to automatically build up the dependency relationship in the process. And based on the dependency relationship, some advanced analysis features on the process would be proposed to help and guide user to optimize and manage the process.

The rest of this paper is organized as follows. In section 2, we will introduce the process model we adopted and the formal description of the dependency relationship. And in section 3 the approach detail to automatically build up the dependency relationship will be given as well as related algorithms will be illustrated. In the section 4 we will briefly depict the architecture and implementation of the tool based on the approach we proposed. And a case study about the testing service process will be illustrated in the section 5. And the related works and conclusion will be summarized in the section 6 and 7 correspondingly.

II. APPROACH OVERVIEW

A. Process Model

Currently, there are many kinds of process model in the process modeling domain [6][7][8]. In this paper, we would like to propose a general process model to present the service delivery process as shown in the figure 1. To simplicity, the elements we would like to involve into this simplified model

Manuscript received March 6, 2010.

Feng Liu, Qian Ma, Hao Wang, Liang Liu, Ying Chen are with the IBM Research – China, Beijing, 100193, China. e-mail: {liufcrl, maqian,wanghcrl,liuliang,yingch}@cn.ibm.com.

Krishna Ratakonda is with the IBM Thomas J. Watson Research Center, Hawthorne, New York, US. (e-mail: ratakond@us.ibm.com)

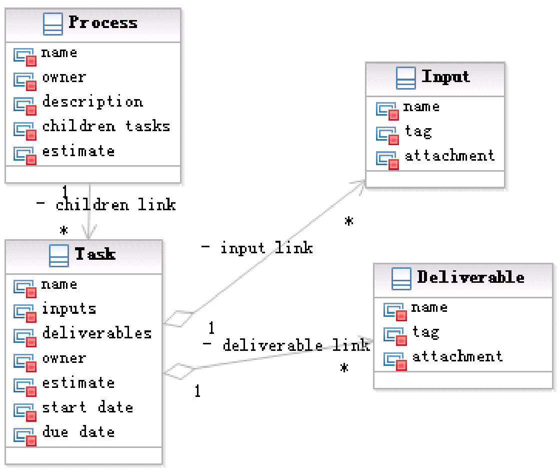

Fig. 1. A Simplified Process Model

include process, task, input and deliverable. Other elements which are irrelevant to the dependency analysis have been ignored in this simplified model.

1) Process

Process is the root model to present whole process. It includes some common attributes such as process name, owner and description. In addition to these basic attributes, the process model should know which tasks are under it. So a children task list attribute is added to process to present the total tasks in this process.

2) Task

Task is the basic element in this process model to present the basic operation unit. It defines the works which has been assigned to a particular team member and the beginning and end date of this task. In addition to these basic attributes, we also define the two types of link in task to present the input and output relationship which correspondingly link to the input and deliverable artifact.

The type of input link presents to the input artifacts which are required by the task. The inputs should be ready before the task can be started.

The type of deliverable link presents to the output artifacts which are delivered by the task. These deliverable delivered by this task at same time are the inputs to the downriver tasks.

3) Input

Input is a type of artifact which presents the required artifacts by the task. Generally, there are initial inputs from the external system to the process to begin this process. And other internal inputs into tasks are also the deliverables by the previous tasks. The key attributes in input model are tag and attachment. The attachment is the real document or any material of the artifact. For example, the input into the task “Write testing design document” maybe an empty testing design template document. After processing by the task, it should deliver a completed testing design document as its deliverable.

To help to build the task dependency based on the inputs and deliverables, we introduce “tag” mechanism. The functionality of the “tag” will be illustrated in following section.

4) Deliverable

Deliverable is a type of artifact which presents the output by the working task to other task. The attributes set in deliverable are same as input as described above.

B. Dependency Formal Definition

In our tooling platform, the process is abstracted as a model tree such as work breakdown structure (*WBS*). But from the abstract perspective, the process is consists of tasks and is constructed as directed graph.

Definition 1: Given that t_x represents one task in the process, the tasks and their dependencies relationship are depicted as a directed graph while each task is represented as vertex and the intra-task dependency as the directed edge. The directed graph Process Graph (*PG*) is defined as:

$$PG = (V, E) \text{, where } V = \{t \mid t \in all\ tasks\},$$

$$E = \{<t_u, t_v> \mid t_u, t_v \in V \wedge t_v\ depends\ on\ t_u\}$$

PG is basic abstract process diagram without weight information. We would define more completed process diagram with weight information on the edge future.

Definition 2: Given that *PG* and t_v and t_u represents two tasks which have dependency relationship in *PG*, and w_{vu} represents the weight of the edge from t_v to t_u which the weight value is equal as the estimate of the t_v. The weighted directed graph Weighted Process Graph (*WPG*) is defined as:

$$WPG = (V, E_w) \text{, where } V = \{t \mid t \in all\ tasks\},$$

$$E_w = \{<t_u, t_v, w> \mid t_u, t_v \in V \wedge t_v\ depends\ on$$

$$t_u \wedge w\ is\ the\ estimate\ value\ of\ t_u\}$$

WPG is *PG* with weights information on the edges. It will be used for analyzing some advanced features on the process.

Definition 3: A task dependency path (*TDP*) is defined as:

$$\forall t_u, t_v \in V \text{, if } \exists t_1, t_2, \ldots, t_n \in V \text{, then}$$

$$TDP = (t_u, t_1, t_2, \ldots, t_n, t_v), where <t_u, t_1> \in E \wedge$$

$$<t_1, t_2> \in E \wedge \ldots \wedge <t_n, t_v> \in E$$

TDP represents a task link chain in *PG* which can be either with weight or without weight information.

Definition 4: Convert *PG* to an undirected process graph (*UPG*). A connected component of *UPG* is a sub-graph in which any two vertices are connected to each other by paths. This connected component is defined as a Connected Sub-Graph (*CSG*). All *CSG* of the *PG* should satisfy:

$$PG = CSG_1 \cup CSG_2 \cup \ldots \cup CSG_n$$

Definition 5: A Critical Task Path (*CTP*) in *WPG* is defined as:

$$CTP = TDP_i \text{, where } TDP_i \text{ has}$$

$$\max\Big(\sum_{t_i, t_j \in V_{TDP_i}, <t_i, t_j> \in E_{TDP_i}} w_{ij} \Big).$$

III. Dependency analysis

A. Build Dependency Relationship in Process

Before the dependency relationship has been built up, the process graph only include tasks nodes and related inputs and deliverables but not link among the tasks. We propose a method which leverages the inputs and deliverables artifacts to analyze the task dependency in the process.

The meaning of task dependency is that the downstream task needs some deliverables from its previous task as its input. So these two tasks should be built a link to denote the dependency link in the *PG*. In the process, there are lots of artifacts as inputs and deliverables and related with various tasks. So how to identify which input and deliverable should be built a link is a key issue.

For example, in the testing service process there are two tasks: 1) review and confirm test strategy and 2) determine build strategy. The owner of the task 1 should review and approve the test strategy document if this document is meeting requirements. After the task 1, the task 2 should take the approved test strategy document as input and write the master test plan document via referring approved test strategy document. This relationship is shown in figure 2.

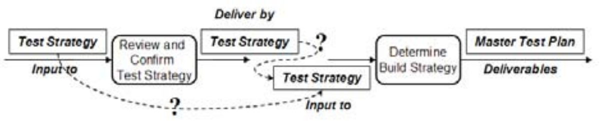

Fig. 2. Sample Tasks with Inputs and Deliverables

In this case, we should identify two test strategy artifacts: one is deliverable of the task 1 and other is input of the task 2. These two artifacts contribute a logic relationship between these two tasks – dependency. But there are probably many artifacts to present various status of the "test strategy" in the whole process either as inputs or deliverables. We need to know which one is correct artifact to set up the dependency link.

To identify correct pair of the artifacts, we would introduce the "tag" mechanism within inputs and deliverables. The "tag" is an additional attribute of the input and deliverable. It represents the status and stage of the artifact. Every input and deliverable should be attached a "tag" in addition to artifact name. We leverage the combination of the tag and the artifact name to identify correct pair of the input and deliverable.

Back to our above sample, the process would be changed as figure 3.

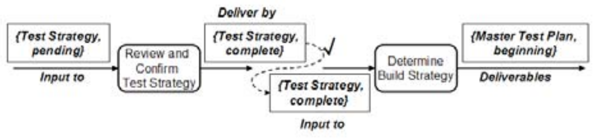
Deliver by
{Test Strategy, pending}
Input to
Review and Confirm Test Strategy
{Test Strategy, complete}
√
{Test Strategy, complete}
Input to
Determine Build Strategy
{Master Test Plan, beginning}
Deliverables

Fig. 3. Sample Tasks with Tagged Inputs and Deliverables

According to combination of the artifact name and the tag, we can setup a link between these two tasks and build up the dependency in *PG*. Related algorithms will be illustrated in next section.

B. Related Algorithms

1) Dependency Analysis

Based on the definition above, an algorithm to analyze the tasks dependency in the process can be obtained. The pseudo-code of the algorithm is listed as follows.

```
Input: $V$ , all inputs and deliverables artifacts
Output: $E$
Algorithm: Dependency_Analysis
List inputSet = new List();
List outputSet = new List();
For each vertex t in V {
    For each t.input in t
        Put (t.input, t.id) into inputSet;
    For each t.deliverable in t
        Put (t.deliverable, t.id) into outputSet;
}
For each inputRecord in inputSet {
   For each outputRecord in outputSet {
      If(inputRecord.first.name == outputRecord.first.name
 &&inputRecord.first.tag==outputRecord.first.tag){
            Task tu = inputRecord.second;
            Task tv = outputRecord.second;
            Put <tu, tv> to E;
      }
   }
}
Return E;
```

This algorithm takes a complex data structure to store input or deliverable artifact and its related task. Every node in this list is constructed one structure of the artifact and its task. After find a correct pair of input and deliverable, the algorithm construct a link using two tasks related with input and deliverable and put this link to the edge set *E*.

2) Build up WPG

Leveraging the edge set *E* we can build up the *WPG* which is the abstract process graph with weight information on the edge.

```
Input: $PG = (V, E)$
Output: $adjMatrix(WPG)$
Algorithm: Buildup_WPG
Int[][] adjMatrix = new int[][];
Initialize adjMatrix;
For each edge <tu, tv> in E {
    Int w = tu.weight;
    adjMatrix[u][v] = w;
}
Return adjMatrix;
```

In this algorithm the adjacency matrix is used to present the process graph. Every element on the position (u, v) in the adjacency matrix represents the weight of the link from t_u to t_v. The value of the weight shows that the estimate effort to finish the task t_u. If there is no edge between t_u and t_v, the element in the adjacency matrix is 0.

3) *Critical Tasks Path (CTP) Analysis*

Base on the *WPG* and definition above, several value-added analysis features can be performed. *CTP* analysis is a valuable feature on the business process management to tell user what is total estimate effort and which tasks are critical nodes. It can guide user to optimize critical tasks to reduce total effort of the whole process.

```
Input: $WPG = (V, E_w)$
Output: $CTP$
Algorithm: Critical_Task_Path_Analysis
List CSGSet = new List();
List CTPSet = new List();
CSGSet = Compute_All_CSG (WPG);
For each CSGi in CSGSet{
    CTPi = Compute_CTP(CSGi);
    Put CTPi into CTPSet;
}
Int maxEffort = 0;
For each CTPi in CTPSet {
    If(CTPi.totalEffort > maxEffort)
        targetCTP = CTPi;
}
Return targetCTP;

Compute_All_CSG (WPG)
   For each task in V {
       Vi = all reachable vertices start from task
       $Ei = \{<tu,tv> | tu,tv \in Vi \wedge <tu,tv> \in E\}$
       Put CSGi=(Vi, Ei) into CSGSet;
   }
   Return CSGSet;

Comput_CTP(CSGi)
  List CTP = new List();
  For each t in $V_{CSGi}$ {
    st = compute earliest start time for t;
    lt = compute latest start time for t;
    if(st == lt)
       put t into CTP;
  }
  Return CTP;
```

Normally, the process only has one connected component. But this algorithm assumes a more general case in a process diagram: it may be partly disconnected. So the algorithm tries to find all connected components firstly and compute *CTP* for every *CSG*. Then it is according to compare the total estimate effort for every *CSG* to find the *CTP* which has max total estimate for the whole process. And the task nodes on this *CTP* are critical tasks for the process.

IV. PROCESS MANAGEMENT TOOL

A. *Architecture*

A process management tool has been developed based on the Eclipse platform. Leveraging this tool the user can input and build up the process, analyze the dependency within the process, optimize the task distribution and calibrate the task workload.

The architecture of the process analysis component is depicted as the figure 4 below.

The user should provide all tasks and inputs and deliverables information into the system via input UI. The process dependency graph will be automatically built up after tasks dependency analysis. Based on the process graph and

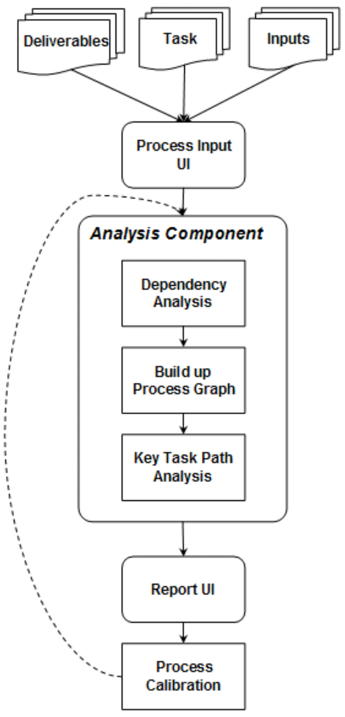

Fig. 4. Architecture of the Process Analysis Tool

estimate information, the critical tasks path can be calculated and reported in a graphic UI. It can guide user to calibrate and optimize the process to improve the efficiency and save more efforts.

B. *Tool Implementation*

In detail on the implementation of the tool, we build a group of plug-ins based on the Eclipse platform. The process model has been defined using EMF and a process work breakdown structure editor is provided for inputting tasks information and creating input and deliverable artifacts.

In addition to input process UI, the tool also contributes a menu action UI to trigger analysis component and a report UI to show the analysis results. Based on the report UI, user can check critical tasks path and calibrate the estimate for every task according to the analysis results then recalculate.

Both UI and analysis component are implemented using Java JDK 1.5. SWT and BRIT are used for implementing report UI. Some screenshots of the tool will be shown in case study section as follows.

V. Case Study

A. Case Scenario

We would like to use the software testing service process as an example to illustrate how the tool works on analysis for the process. The software testing service process include many tasks which cover every testing phases such as test planning, test design and test execution. The test artifacts such as test strategy document, test design document, etc., have been as inputs and deliverables and flow in the process.

The left-top section is the process management explorer. The user can create new process and trigger the analysis action on specific process. Also the process analysis tool provides an editor to create tasks with inputs and deliverables. After creating these tasks elements, the tool construct the process as a work breakdown structure in the process editing area as shown in figure 5 below.

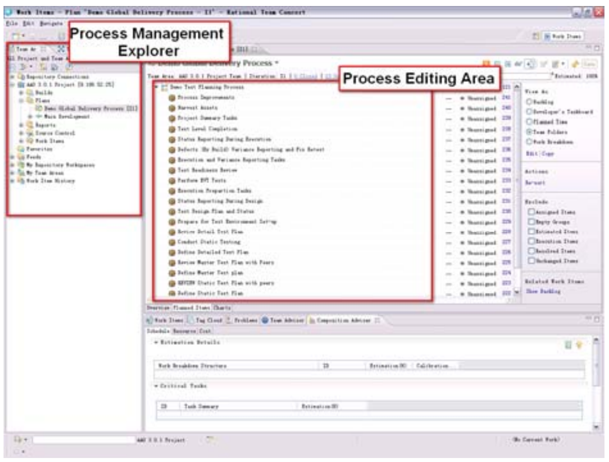

Fig. 5. Process Editing UI to Input Tasks

After tasks information preparation, we can start the dependency analysis to build up the relationship in the process and execute more advanced analysis features on the process. The result graph of the dependency analysis is shown in figure 6.

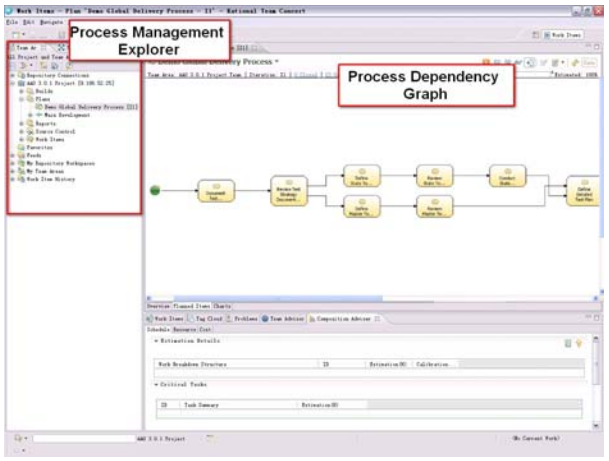

Fig. 6. Result Graph of Process Dependency Analysis

B. Advanced Analysis Features

Based on the process graph analysis, the tool also provides the functionality to analyze critical task and the critical paths. The analysis result will be shown in report UI component as follows figure 7 and figure 8.

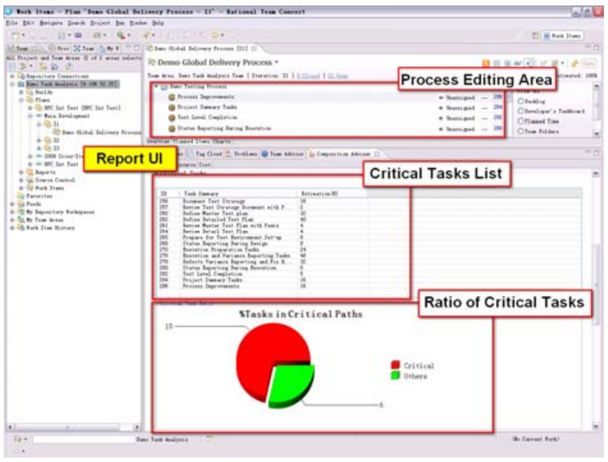

Fig. 7. Report UI on Critical Tasks

This analysis report shows that the critical tasks list and the ratio of the critical tasks in total tasks. In the task list, user can calibrate the estimate for every task and the updated result can be recalculated and displayed automatically. Following the critical task list view, a report pie chart presents how many critical tasks in the total.

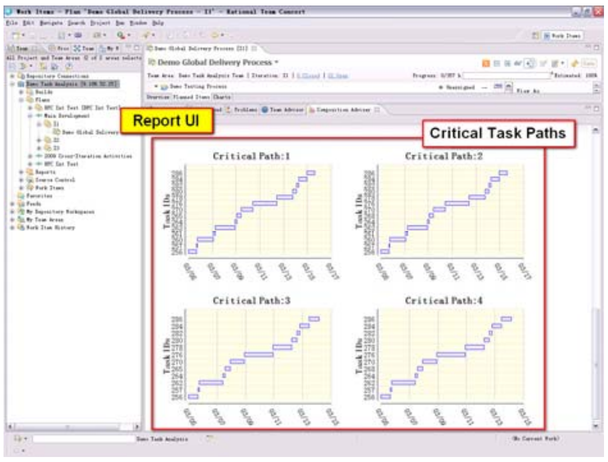

Fig. 8. Report UI on Critical Task Paths

The diagram in figure 8 shows the critical task path using Gantt chart. Because there may be not only one critical path in a process, this view gives all critical paths in the process. The x-coordinate represents date and the y-coordinate represents the task id. Also the length of the task node presents its estimate.

VI. Related Works

There are several research approaches on process construction which our works is related. The famous process frameworks to guide construct process are RUP and OMG. These process frameworks provide a group of methodology and guidance to model the process in a conceptual level.

Other more detail level approach to construct the process is a scenario-driven way proposed in [11]. And an industry process construction approach is introduced by Rachel Cooper in [12]. These methodology and approaches either provide guidance on modeling process or detail method to construct the process.

With respect to the dependency analysis, a main research direction is focusing on the fault injection. This approach captures the dependency in distributed system via injecting fault into the related components. The advantage of this method is that it has ability to dynamically discovery dependency. But in service delivery process, there is no evidence dependency among tasks like as application components in distributed system. We should trace the artifacts flowing in the process to build up the dependency. So fault injection method hardly applies into dependency analysis in the service delivery process.

VII. Conclusion and future works

In this paper, a model-based dependency analysis approach for service delivery process has been proposed. This method can automatically build up the dependency relationship among tasks and help to improve the efficiency and quality of the service delivery process. Based on the dependency analysis, some advanced analysis features on the process also been proposed to guide user more clearly understand which tasks are critical in the process and how to reduce time cost for whole process.

Several extension directions can be planned in future works. First of all is about the model extension. In this paper we just propose a simplified process model to present the process. Actually, in a service delivery process should include more complex level model to present the real business requirements. For example, we should consider separating process into several phases instead of directly to tasks. Because most service delivery processes include more than one stage which includes a group of tasks. Divide process to multiple levels model is more close to real process. The related issues are how to group the task to the phase and how to build up the dependency among phases according to the tasks dependency.

Another is that how to optimize the algorithm to build up the dependency more effectively. Especially if we extend the model of the process to multiple levels, we should consider improving the algorithm to enable it to adapt more complex process model.

References

[1] N Seth, SG Deshmukh, P Vrat, “Service Quality Models: A Review”, International Journal of Quality & Reliability Management, Vol. 22 No 9, 2005, pp. 913-949

[2] W.M.P. van der Aalst, A.H.M. ter Hofstede, and M. Weske, “Business Process Management: A Survey”, Proceedings of the 1st International Conference on Business Process Management, volume 2678 of LNCS

[3] Saurabh Bagchi+, Gautam Kar, Joe Hellerstein, “Dependency Analysis in Distributed Systems using Fault Injection: Application to Problem Determination in an e-commerce Environment”, 12th Intl. Workshop on Distributed System: Operation and Management DSOM’2001 Nancy France, October 15-17, 2001

[4] A. Brown, G. Kar, A. Keller, “An Active Approach to Characterizing Dynamic Dependencies for Problem Determination in a Distributed Application Environment,” IEEE/IFIP International Symposium on Integrated Network Management, pp. 377-390, 2001.

[5] Alexander Egyed, “A Scenario-Driven Approach to Trace Dependency Analysis”, IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 29, NO. 2, FEBRUARY 2003 1

[6] Pnina Soffer, Yair Wand and Maya Kaner, “Semantic Analysis of Flow Patterns in Business Process Modeling”, Lecture Notes in Computer Science, 2007

[7] Recker, Jan C. and Rosemann, Michael and Indulska, Marta and Green, Peter, “Business Process Modeling: A Comparative Analysis”, Journal of the Association for Information Systems, 10(4). pp. 333-363.

[8] Alexander Dreilinga, Michael Rosemannb, Wil M.P. van der Aalstc, and Wasim Sadiq, “From conceptual process models to running systems: A holistic approach for the configuration of enterprise system processes”, Decision Support Systems Volume 45, Issue 2, May 2008, Pages 189-207

[9] Theodora Kourti and John F. MacGregor, “Process analysis, monitoring and diagnosis, using multivariate projection methods”, Chemometrics and Intelligent Laboratory Systems Volume 28, Issue 1, April 1995, Pages 3-21

[10] Graham Winch, “Models of manufacturing and the construction process: the genesis of re-engineering construction”, Building Research & Information, Volume 31, Issue 2 March 2003 , pages 107 – 118

[11] Julio Cesar Sampaio do Prado Leite, Graciela D. S. Hadad, Jorge Horacio Doorn and Gladys N. Kaplan, “A Scenario Construction Process”, Requirements Engineering, Volume 5, Number 1, 2000.7.

[12] Rachel Cooper, Michail Kagioglou, Ghassan Aouad, John Hinks, “The Development of a Generic Design and Construction Process”