A Resource Based Formal Methodology to Detect Non-Conformance in Processes

Sean Thompson[1] and Torab Torabi[2]

[1] S. Thompson, Computer Science Department, La Trobe University, Melbourne, Australia, e-mail: sean@nostin.com
[2] T. Torabi, Computer Science Department, La Trobe University, Melbourne, Australia, e-mail: T.Torabi@latrobe.edu.au

Abstract – **The methodology described in this paper seeks not so much to break away from the other generic, broad methodologies presented in the literature thus far, but rather to apply our concept to a small and easier aspect of the process – resources. By applying our model to just process resources, we can keep it a lot simpler and easier to understand. Having said that, our methodology still remains generic and applicable to processes across all domains instead of being targeted at one, like manufacturing or software engineering the way most other presented models do. The model we describe here is specifically designed to detect instances of non-conformance between a "prescribed" process and its "observed" process counterpart by structuring the data of the two and then making comparisons. A mathematically formal model is presented with a small illustrative example of how it can be usefully applied.**

Index Terms **– process non-conformance, process deviation, process resources, resource non-conformance, resource deviation.**

I. INTRODUCTION

We aim here to provide a formal, generic model which details the methodology we have developed to detect *resource* based non-conformance between a process prescription and its observed enactments. Detecting non-conformance early in a process can have several benefits to an organization, including curbing the consequences of a potentially dangerous process deviation, discovery of improved ways of doing things, prevention of precarious situations and achieving objectives in a more efficient, effective and timely manner. A generic approach akin to the model we present in this paper has not so far been presented in the literature. We have endeavoured therefore, to keep our approach generic and formal in here so that our methodology can be applied to as wide a range of processes as possible in many different domains – not just software or business processes.

We begin, in this paper, by making an important distinction between the *process resource* domain and the *process activity* domain within a prescribed process and we treat these two domains completely separately when applying our model to a process. This is a distinction that many researchers working on process non-conformance or "deviations" which has been conducted (particularly onward from the early 1980's [2, 3]). The beginnings of this research were presented in [4] and [5] however these publications did not detail the aforementioned distinction between the resources and activities. In this paper we focus solely on explaining our approach to detecting non-conformance between the *observed* process and the *prescribed* process by examining only their associated resources. To our knowledge, such an examination based on resource non-conformance has not been presented in the literature before. Two immediate benefits apply when taking this manner of approach:

1. The examination of the state of processes tangible assets both before the process has been enacted and then afterwards is a great indication of how the process has performed. In most cases, the process will have some specific goal which uses these resources in the course of its enactment. The process will produce some outcome based on the resources it has used which should be easy to observe.

2. It is practically a lot easier to observe the changes in something tangible like resources which can be seen and touched and counted than it is to discern in activities, where interpretation of a persons actions is required for comparing them to a set of prescribed actions.

Considering the above, the theoretical model we present in this paper is concerned with the detection of process non-conformance, with a specific application to the resources involved. Aside from our body of research, the work presented in this area is broad and covers a wide range of methodologies. The approaches most directly related to our own are the *process discovery* style approaches such as the frameworks discussed in [7] and [8] where the process model is "discovered" by examining actual event data. The discovered model is then compared with other process enactment data to detect discrepancies, which should theoretically be easy to do because the comparison data from the discovered model should be on the same level as the enactment data. Methodologies based on fuzzy logic/fuzzy sets theory have also been proposed such as the approach described in [10] where the authors create a simulated "flawless" execution of the process which is then compared to actual enactments. By the authors own admission in this instance, a flawless execution of almost any process seems very unlikely, perhaps impossible which would almost guarantee non-conformance in every enactment of the process regardless of how well it was performed.

Davenport in [17] defines a process as a "set of logically related tasks formed to achieve a defined … outcome". It may, depending on the domain in which it is enacted, include machines, methods, rules, organizational structures, sub-procedures and computerized tools to aid in achieving its goal(s) [18, 19]. In relation to our model, we treat a process as a set of activities (performed by *actors* – being either human or automated [20] which are responsible for their enactment [12]) and each activity within the process may have a set of resources associated with it as will be further explained in the formalization in section 4. We also assume as in [7] and [11] that the activities defined in a process may not always be performed in a sequential fashion, but perhaps simultaneously, overlapping or in parallel. Also in our model, although resources are treated independently and are related to the process, each activity within the process owns a separate relationship with each resource it uses. The *usage* of each resource per activity is hierarchically structured underneath the associated activity and treated separately. This is also explained further in section 4.

Since other bodies of work in this area make reference to concepts such as "inconsistency" or "deviation" as well as non-conformance, what we are actually referring to in this paper when we mention *non-conformance* is an instance where the *observed process* [6] has acted (or neglected to act) in any way which is not in concordance with its prescription. Incidentally, the terms *deviation* and *inconsistency* we consider being sub-types of non-conformance the same way they were initially distinguished in [1] where *inconsistencies* relate to discrepancies relating to states or activities within the process whereas a *deviation* is a concept relating to transition between them (activities/states). As we do not cover transition between activities in this methodology, deviations are not detailed in this paper, only resource based inconsistencies.

The way resources are defined in the applicable literature is also not consistent, requiring us to specify exactly what we are referring to when we mention *resources* (also covered in more detail in section 2). Some of the methodologies presented in the literature assert that resources are even required by process activities in order to be enacted [11] and some (also in [11]) contend that all manner of process effects such as humans and time may be considered a resource. In [15], Hu et al suggest that humans, machines and applications could be resources and also, incidentally, that resources may carry out tasks and activities themselves. In our approach, we do not consider all these things resources. Only the tangible assets that are directly used by the process are considered resources, these being the subject of the actions applied to them in the course of enacting activities.

Conversely from the broad array of non-conformance methodologies in the literature not focusing on resources, there is also a significant body of work in the literature on process resources but none of the work is directed at detecting non-conformance. The main body of work in this domain on process resources tends to lean towards areas (among others) such as efficient allocation of resources like the model presented in [13] or flexible allocation of limited resources over varied dynamic demands such as in [14]. There has been methodologies presented on ascertaining the value of resources to a process (or processes) such as the approach detailed on IT outsourcing by Roy et al in [16] and even the experimental paper published in [9] on the deliberate limitation of resources in order to force people to discover more efficient methods of utilizing the insufficient yet available resource supply to achieve the process goal.

The rest of this paper is split into the following sections: in section 2 we specifically define what a resource is and how it relates to the non-conformance model we have developed. Section 3 details the issues faced with resource based process non-conformance detection and the ways we have gone about resolving these issues. Section 4 details the mathematical formalization of the model and section 5 concludes the paper and lays out our plans for future research in this area.

II. Resource Scope

When we refer to a *resource* in this paper, we mean something tangible that has been defined as a resource in the scope of our model for a particular process. In order to confine this definition to something concise, understandable and applicable in this scope, we omit concepts that may be considered resources in other bodies of related research such as in [11] or [15]. We however keep what we consider quantifiable and observable tangible process assets that are defined by a process administrator using our defined model. Concepts such as time (both duration and relative – days of the week, time of day) or human actions and skill sets are some of the concepts (also mentioned in our related application which is presented in [21]). These may constitute being a "resource" in other related models which we omit from our model on resources and instead take care of in our complementary model on process activities [4].

The reason constraining our resource definition as such is because we have attempted to keep resources as a sphere within the process as loosely coupled from other spheres (such as activities) as possible. The mathematical formalization of how we define a resource is in the coming section 3 where we break down and formalize exactly what must and can be specified when defining a resource in a process using our methodology.

III. The Model

In this section, we start by explaining what we define as a process in the scope of this model and how it is mathematically formalized, with the way resources are defined and formalized, and how they relate to the process. The process we are formalizing in this section refers to the *prescribed* process. The *observed* process and the comparison and non-conformance detection methodology are covered in section 4. We will begin then, by mathematically defining the key parameters in our model which we define as follows:

Let P denote a *process.*
Let A denote an *activity.*
Let R denote a *resource.*

A. The Process

As mentioned previously in section 1, the *process* is for the purpose of this formalization kept singular. We do not deal with multiple processes in this paper so we can keep the process denoted simply as P in this case. A process is made up of a set of *activities*. The activities are separate tasks performed by the actors responsible for them [20] and are directly related to the process. In the theoretical scope of this model, each activity only exists under the process it is defined under. Thus, we can formalize the process as being:

$$P \equiv \{A_1, A_2, \ldots, A_i\}$$

Where *i* is the number of activities defined for the prescribed process.

B. The Activities

The way activities are performed in a process is complex to model and is taken care of in some of our previously presented [4] and ongoing research. Although we consider process activities to incorporate such attributes such as responsible actors, human skill sets, modular tasks, time constraints, repetition constraints, sequencing constraints and transitional rules, these are omitted from this paper in favour of simplicity since it does not directly relate to our methodology on resource non-conformance detection. For this reason, in this instance we will confine process activities to the following formal definition:

$$A_i \equiv \{R_{i_1}, R_{i_2}, \ldots, R_{i_k}\}$$

Where $A_i \in P$ and *k* is the number of resource usage instances prescribed for activity A_i.

C. The Resources

Technically, a resource is autonomous and independent of the process or processes that use it. A resource can be defined and used by multiple processes and can exist outside a process prescription/definition unlike process activities. With this in mind, it is the *resource relationship* between the resource and its associated *process activity* which we observe and compare in this methodology. The way this relationship behaves affects the usage and quantities of the related independent resource. Resources therefore, have certain characteristics associated with them and must subscribe to the following constraints:

A resource must be exclusively either *exhaustive* or *non-exhaustive*, and if it is exhaustive, it must specify whether it is one of *generative* or *consumable* or whether it is both or neither. Basically, this refers to whether the resource is quantifiable or not. Using an example we illustrated in [21], when cooking a pizza an example of an exhaustive resource might be the cheese we sprinkle onto the pizza because it is quantifiable and limited whereas the oven the pizza is cooked in is not exhaustive in this scope because once the oven is used, the same oven remains available for immediate reuse again. The *generative* and *consumable* tags refer to an exhaustive resource and whether we are generating more of the resource or consuming some/all of it and therefore leaving less. Incidentally, a resource which is considered non-exhaustive in one case or process may just as easily be considered exhaustive in another – for example an oven manufacturing process. It is for this reason that the *relationship* the resource has with the process activity is what we use for comparison when testing for non-conformance. It is the relationship the oven resource has with the pizza making process.

With that said, let us specify whether a prescribed resource is exhaustive or not by denoting its type as E where E denotes an exhaustive type and $\overline{E}$ denotes a non-exhaustive type.

Now if the resource in question is exhaustive E typed, then whether it is *generative* or *consumable* to the process activity it is related to must be specified. A specification must be given for both its generative and consumable status. These are formalized as follows:

Let G denote the generative status the process activity has with the exhaustive resource. These may be one of two types where G denotes that the process activity is *generating* the resource and $\overline{G}$ denotes that the process activity is not *generating* the resource.

Let C denote the consumable status the process activity has with the exhaustive resource. These may be one of two types where C denotes that the process activity is *consuming* the resource and $\overline{C}$ denotes that the process activity is not *consuming* the resource.

Furthermore to this concept, we also introduce two optional boundary value parameters to resources which are exhaustive types. That being, we can optionally define a minimum boundary that constrains the smallest amount of a particular resource that should be consumed or generated by the process activity it is related to and also a maximum boundary limit. The reasoning behind making these limits available to someone prescribing a process is so they can make an informed decision on how much is too much or how much is too little. If these boundaries are exceeded, a non-conformance instance is observed. Both these parameters are optional and specifying one without the other is legal in the scope of our model. Some thought and foresight is therefore required by anyone seeking to impose these boundaries when prescribing the process. So,

Let $g_{\min}$ denote the optional *minimum* boundary placed on exhaustive typed resources to be *generated* and let $g_{\max}$ denote the optional *maximum* boundary limit.

Let $c_{\min}$ denote the optional *minimum* boundary placed on exhaustive typed resources to be *consumed* and let $c_{\max}$ denote the optional *maximum* boundary limit.

Tying these prescription rules together, we arrive at the complete formalization for *prescribing* the relationship an available *resource* has with a *process activity* and the constraints for how this resource is to be used. This is formalized in complete form as:

$$R_{i_k} \equiv \{\overline{E} \mid E\{\overline{G} \mid G\{[g_{\min}],[g_{\max}]\}, \overline{C} \mid C\{[c_{\min}],[c_{\max}]\}\}\}$$

Where the '|' symbol refers to exclusive "OR" and the square brackets "[ ]" refer to optional parameters.

IV. Making The Comparisons

This section details how observed data from process enactments is formalized and related to the process prescription and also how the comparisons are made in order to detect any instances of resource based non-conformance between the observed process and its prescription. These comparisons are made via two comparison rule flows which are shown in figures 1 and 2, which incidentally are also both borrowed from our paper presented in [21].

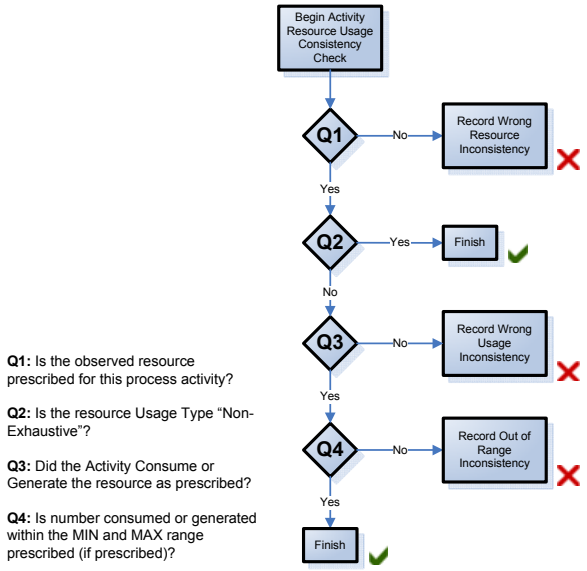

Fig.1 Comparison Rule Flow 1

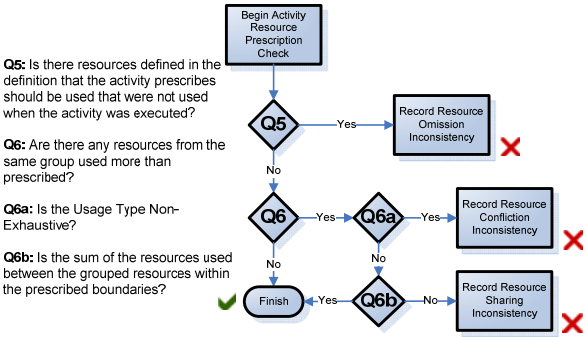

Fig.1 Comparison Rule Flow 2

A. The Observation

Before we can begin to compare the *observed* process to the *prescribed* process, we need to formally define the observed instance. We differentiate the prescribed from the observed by referring to any observed process data using the "hat" ^ notation. Therefore we let $\hat{P}$ represent an observed instance of the prescribed process P. It also follows therefore that $\hat{A}_i$ is an observed instance of the prescribed activity A_i and that $\hat{R}_{i_k}$ is an observed instance of the resource R_{i_k}.

B. Checking the Relationship Existence

This relates to the first step in both flows one and two and refers to checking the existence of a relationship between the observed resource usage and the prescribed resource usage. In the first flow, we are checking that for every observed resource $\hat{R}_{x_y}$ there exists a prescription R_{x_y}. Here we are ensuring that every resource used has at least a corresponding prescription for its usage (for that activity). This type of non-conformance is referred to as a *wrong resource inconsistency* as shown in figure 1. In the second flow, we are simply doing the opposite and checking that for every prescribed resource usage R_{x_y} there exists an observed usage $\hat{R}_{x_y}$. In other words, we are ensuring that for every resource prescribed to be used for a given process activity, we have not missed any when enacting. This type of non-conformance we refer to as being a *resource omission inconsistency* as seen in figure 2.

C. Checking the Type

Now that we know the relationship exists between the prescription and the observed, we must ensure it is the correct type. In Q2 of flow 1 (in figure 1), we must now check that the resource was consumed and/or generated as prescribed. However, this only applied if the resource is prescribed as being exhaustive. So, if the prescribed resource is of type exhaustive – being E, then we check the resource generation and consumption status of the observed resource usage.

For example, if there exists $\hat{G} \in \hat{R}_{x_y}$ then there must also exist a corresponding $G \in R_{x_y}$ which also obviously applies to consumption also. In other words, if the prescription states that the process activity should generate a resource and it does not, or that it should consume a resource and it does not (or vice-versa) then we record an instance of process non-conformance. We refer to this type of non conformance as a *wrong usage inconsistency*.

D. Exhaustive Range Checks

The final check (Q4) in flow 1 (figure 1) we make only if the previous checks pass and also if the optional

boundary limit has been defined. So, if the observed resource has been confirmed as being exhaustive type and if the resource was generated and consumed as was prescribed, then we lastly make certain that the amount generated or the amount consumed was both above the specified minimum (if prescribed) and also below the specified maximum (also if prescribed). So, for the observed enacted process, we define two more parameters:

Let $\hat{g}$ denote the amount of the resource observed to be *generated* during the enactment of the process; and

Let $\hat{c}$ denote the amount of the resource observed to be *consumed* during the enactment of the process.

In order for this test to pass, the following checks must all be true:

If there exists $[g_{\max}]$ then $\hat{g} < g_{\max}$ must be true.

If there exists $[g_{\min}]$ then $\hat{g} > g_{\min}$ must be true.

If there exists $[c_{\max}]$ then $\hat{c} < c_{\max}$ must be true.

If there exists $[c_{\min}]$ then $\hat{c} > c_{\min}$ must be true.

If any of these checks fail, a non-conformance instance is observed and recorded and we refer to this type of non-conformance as an *Out of Range Inconsistency*.

V. Advanced Non-Conformance Checks

Consider checks Q6, Q6a and Q6b in the second flow illustrated in figure 2. These checks refer to the additional measures we have taken in this model to guard against what we refer to as *shared* and *conflicting* resource non-conformance. We consider these final checks separately because they require not only additional comparisons but also a slight change to the resource prescription model we finalized previously in section 3.

A. Conflicting Resource Inconsistencies

This non-conformance type refers only to resources which are prescribed as non-exhaustive. This is best described using an example so we will use the previously mentioned "pizza cooking" example from [21] to do so. Suppose a non-exhaustive type resource such as an oven. Perhaps in the pizza parlour there are multiple available ovens. A conflicting resource inconsistency would arise when say the observed activity to cook the pizza would use more than one oven.

The problem is however, that it may be perfectly legal to use either oven, so we need a mechanism of checking to see if more than one of the same types of resources has been used. To accomplish this, we need to modify the resource usage prescription to include *resource groups* for the particular process activity. These are covered in detail in sub-section C.

B. Shared Resource Inconsistencies

Similarly to conflicting resource inconsistencies, this refers to multiple types of the same sort of resource being used in a process and therefore relies on the additional *resource group* formalization added to the resource prescription model. Here however, we refer to exhaustive resource types. Suppose now, that two different types of cheese have been sprinkled on the pizza. Both types are perfectly legal on their own, but collectively how much of each can be used? We must also here provide a mechanism for counting collectively how much cheese from both types has been used and compare the result against the prescribed limits (if they have been specified).

C. Resource Group Types

To account for subsections A and B, we can amend our previous formalization for prescribing a resource to incorporate a *resource group*. By this we mean for every process activity which prescribes a resource usage, instead of referencing the resource itself it can reference a *resource group* which contains references to all the applicable resources for the group. For example, in the pizza cooking process, we may define an "Oven" group which would contain a reference to all available ovens in the pizza parlour. We may also define a group called "Cheese Toppings" which could contain all the different cheese toppings which can legally be used in the scope of the prescribed process activity. Let us define *resource groups* to be S for "set" because "G" is already taken for "generative" and we formalize this as:

$$S_{iq} \equiv \{A_i, R_{i_1}, R_{i_2}, \ldots, R_{i_z}\}$$

Where *i* is the activity referenced by the group, *q* is the identifier of that particular group for the process activity and *z* is the number of resources related to that activity which are classified as being in the resource group.

Our new formalization for prescribing the resource has also now changed to incorporate a reference to its associated group and is as follows:

$$R_{i_k} \equiv \{S, \overline{E} \mid E\{\overline{G} \mid G\{[g_{\min}], [g_{\max}]\}, \overline{C} \mid C\{[c_{\min}], [c_{\max}]\}\}\}$$

This formula describes entirely the prescription for a resource-to-activity relationship a process can have with an existing resource, up to the point we have progressed to in our research model. This model will somewhat change over time as we expand the model to incorporate other ways of checking for non-conformance and incorporating our previously implemented framework presented in [21]. Our plans for improving and expanding this research are mentioned in the coming section which concludes this paper.

VI. Conclusion and Future Work

This paper was aimed at formalizing mathematically a body of research which we have been working on for some

time. The methodology formalized is aimed at the detection of non-conformance between a process prescription and its enactments with specific application to the resources involved in the process. We have covered the key issues relating to basic detection of resource based non-conformance such as boundary limits, typing and relationship existence checking but we also expand into more comprehensive checking such as resource sharing and conflicts to ensure our model is as robust as possible. This model is simplified and should be applicable to a wide range of processes across a variety of applicable domains.

The idea of forming a methodology to guard against resource confliction and resource sharing inconsistencies which we covered in this paper provide an excellent base for us to refine this model and further explore additional mechanisms to ensure the methodology is as comprehensive as possible. We have been working diligently to simplify, improve and expand this methodology and enhance our framework to ensure the model is as robust as possible. Also continuing in our future work is the adaptation of this model to some real world processes both simple and complex to fully gauge the potential value our methodology may provide in a variety of different circumstances.

VII. REFERENCES

[1] G. Cugola, E. Di Nitto, A. Fuggetta, C. Ghezzi; "A framework for formalizing inconsistencies and deviations in human-centered systems", *ACM Transactions on Software Engineering and Methodology (TOSEM)*, Volume 5 Issue 3, ACM Press, July 1996.

[2] V.R. Basili, F.E. McGarry, R. Pajerski, M.V. Zelkowitz; "Lessons learned from 25 years of process improvement: the rise and fall of the NASA software engineering laboratory", *Proceedings of the 24th International Conference on Software Engineering*, ACM Press, May 2002.

[3] A. Fuggetta, "Software Process: A Roadmap", *Proceedings of the Conference on The Future of Software Engineering*, ACM Press, May 2000.

[4] S. Thompson, T. Torabi, P. Joshi, "A Framework to Detect Deviations During Process Enactment", *6th IEEE International Conference on Computer and Information Science*, IEEE Computer Society Press, Melbourne, Australia, July 2007.

[5] S. Thompson, T. Torabi, "A Process Improvement Approach to Improve Web Form Design and Usability", *The 3rd Ubiquitous Web Systems and Intelligence Workshop (UWSI 2007) Colocated with DEXA 2007*, Regensburg, Germany, 3-7 September 2007.

[6] K. Mohammed, L. Redouane, C. Bernard, "Processes: A deviation-tolerant approach to software process evolution", *Ninth international workshop on Principles of software evolution: in conjunction with the 6th ESEC/FSE joint meeting IWPSE '07*, ACM Press, September 2007.

[7] M. Huo, H. Zhang, R. Jeffery, "An Exploratory Study of Process Enactment as Input to Software Process Improvement", *International Conference on Software Engineering*, 2006.

[8] J.E. Cook, A.L. Wolf, "Discovering models of software processes from event-based data", *ACM Transactions on Software Engineering and Methodology (TOSEM)*, Volume 7 Issue 3, July 1998.

[9] L.J. Burnell, J.W. Priest, J.R. Durrett; "Reviewed papers: Assessment of a resource limited process for multidisciplinary projects", *ACM SIGCSE Bulletin*, Volume 35 Issue 4, ACM Press, December 2003.

[10] S. Cîmpan, F. Oquendo; "Dealing with software process deviations using fuzzy logic based monitoring", *ACM SIGAPP Applied Computing Review*, Volume 8 Issue 2, ACM Press, December 2000.

[11] Y. Rezgui, F. Marir, G. Cooper, J. Yip, and P. Brandon, "A Case-Based Approach to Construction Process Activity Specification", *Intelligent Information Systems IIS* '97, 8-10 Dec. 1997, pp. 293 – 297.

[12] M. Dowson, B. Nejmeh, and W. Riddle, "Concepts for Process Definition and Support", *Proceedings of the 6th International Software Process Workshop*, IEEE Computer Society Press, Hakodate, Japan, October 28-31 1990.

[13] J. April, M. Better, F. Glover, J. Kelly, M. Laguna, "Enhancing Business Process Management With Simulation Optimization", *Proceedings of the 2006 Winter Simulation Conference*, Monterey, California, USA, December 3-6, 2006.

[14] U. Deshpande, A. Gupta, A. Basu, "Adaptive Problem Solving among Business Organizations through Flexible Resource Allocation", *International Conference on Advanced Computing and Communications (ADCOM)*, 20-23 Dec. 2006 Page(s):382 – 387.

[15] L. Hu, Y. Hu, "Resource perspectives in the context of process management of product development", *7th International Conference on Computer-Aided Industrial Design and Conceptual Design*, 2006. CAIDCD '06, 17-19 Nov. 2006 Page(s): 1 – 5.

[16] V. Roy, B.A. Aubert, "A resource-based analysis of IT sourcing", *ACM SIGMIS Database*, Volume 33 Issue 2, ACM Press, June 2002.

[17] T. Davenport, "Process Innovation: Re-engineering Work through Information Technology", *Harvard Business School Press*, Boston MA, 1993.

[18] A. Blyth, "Business process re-engineering: What is it?", *ACM SIGGROUP Bulletin*, Volume 18 Issue 1, April 1997.

[19] W.A. Florac, A.D. Carleton, *Measuring the Software Process: Statistical Process Control for Process Improvement*, Addison-Wesley, 1999.

[20] A. Caetano, M. Zacarias, A.R Silva, J. Tribolet, "A Role-Based Framework for Business Process Modeling", *Proceedings of the 38th Hawaii International Conference on System Sciences*, Hawaii, USA, 2005.

[21] S. Thompson, T. Torabi, "Towards Formalizing Resource Based Non-Conformance in Business Processes", *19th Australian Software Engineering Conference (ASWEC),* Perth, Australia, March 2008 (accepted).