An Observational Approach to Practical Process Non-Conformance Detection

Sean Thompson[1], TorabTorabi[2]
[1]La Trobe University
Bundoora
Australia
sean@nostin.com

[2]La Trobe University
Bundoora
Australia
T.Torabi@latrobe.edu.au

ABSTRACT: *A key challenge facing process non-conformance detection is to develop approaches not only effective across different domains and for different kinds of processes, but also to provide a practical and easy to implement solution. Since this is a process improvement area, the commercial realities of today dictate that if a solution cannot be implemented easily, quickly and cost-effectively then it is probably not worth implementing at all regardless of its effectiveness. The focus of this paper is on presenting a simple and practical approach to process non-conformance detection. We propose an approach based on specifying non-conformance rules to enhance an existing process specification without modifying it. The rules specified are used to check for conformance of process enactment data. Our methodology is flexible in its instantiation and because it is observatory, can be implemented with minimal interference with the existing process. We evaluate this approach using a case study.*

Keywords: Non-conformance detection, Process specification, Software evaluation

Received: 18 November 2009, Revised 18 January 2010, Accepted 29 January 2010

1. Introduction

There have been varied attempts at detecting process non-conformance presented in the literature. One of the main issues in this area is the lack of a practical approach that can be applied easily and generically with minimal impact on the process itself. We see a need for a solution that can be applied easily and quickly to a process irrespective of how the process has been initially specified. The solution should also be non-invasive, that is to say the process specification should not need to be changed to accommodate the methodology.

A process is a structured set of activities carried out to achieve a business goal. The activities that make up the process may be carried out in a number of different ways, including sequentially, concurrently, simultaneously, overlapping or in parallel [1].

Non-conformance is an instance of difference between a process specification and an enactment of that process. There are two types of non-conformance we recognize in this approach: deviations and inconsistencies. An *inconsistency* is a concept relating to the state of a process or its associated activities whereas a *deviation* is a concept relating to non-conformant transition between process activities. Cugola et al. first introduced this distinction in their paper on formalizing inconsistencies and deviations in human-centred systems [2].

The proposed approach is concerned with the enhancement of a process specification by defining *rule-sets* to complement it. The rule-sets contain rules that specify values, constraints and conditions the process enactment data must comply with. Non-conformance is detected by comparing enactment data with these rules.

1.1 Related Work

Research in this area first began to evolve in its current state following the 1999 paper by Cook and Wolf [3] on mining process event data in order to construct a real process specification from it. This reverse approach at modelling the process from data already recorded from existing enactments spawned an experiment by Huo et al. [4] that used this method to compare the discovered specification to further process enactments. The major problems with this idea is first the sheer amount of data needed to discover the initial process model and second, the fact that many iterations of process enactments need to take place before enough data can be gathered for the discovery. This inevitably means that the process may already be in a relatively mature state before the approach can be implemented. Some types of processes are also simply not enacted often enough to be suitable for this. Taking this into consideration, we devised our own approach to be applicable to a wide range of processes, from the beginning and regardless of how often they are enacted in practice.

In 2000, Cîmpan, and Oquendo [5], published a fuzzy logic approach to non-conformance detection. The idea was to model a "perfect" simulation of how a process should be enacted and set the simulated model as the expectation for further enactments. Actual process enactments were then mapped to the expected behaviour to look for variations with the use of fuzzy sets theory. The authors conceded that the issue with this kind of idea was that the likelihood of a perfect process enactment was realistically close to zero, which virtually guarantees non-conformance of some kind. It also did not seem to address the possibility that processes could be enacted in a number of different yet equally valid ways. Nevertheless, to accommodate the shortcomings of expected perfection, the authors introduced the "tolerable" deviation concept in which they define certain situations in which certain deviations may be overlooked or tolerated by the approach. This concept is one we take a similar viewpoint in our own research with the inclusion of "exception types" regarding our own definition of non-conformance.

More recently, Mohammed et al. [6] presented a similar approach to our own, in which two co-existing process models are utilized. The approach works by using a guidance driven process model to dictate how the process should be carried out, similarly to a process specification. The other model is implemented only to observe the actions of human actors involved in the process. Comparisons are made between the two as the observations are made and recorded and "deviations" are detected as a result of these comparisons. Rule specification is also introduced in this approach but only in the form of determining the relative severity of a detected deviation and whether or not it should be tolerated and the process should be allowed to continue. This approach is therefore not strictly observational in its implementation.

In this approach, the authors consider the concepts of "deviation" and "inconsistency" differently to the way we consider it in our own research and the way it has been previously presented in the literature, such as in [2]. They do not consider the tern "non-conformance", rather they consider a "deviation" to constitute what we would otherwise consider to be non-conformance and that an "inconsistency" refers to abnormal process data resulting from a deviation that has occurred. Furthermore, the authors do not consider the possibility that non-conformance (or deviations in this case) may be instigated for a good reason – the actor has discovered what he or she believes to be a better way of doing things. It is important to recognize this possibility as an opportunity for process improvement and is one of the main reasons why in our own approach a strictly observational implementation is sought.

Finally in 2008, van der Aalst et al. [7] presented an approach in which the conformance of software services is checked in the context of its behaviour as expected by other software services within its environment. Behavioural conformance is tested via an implemented Petri net system from BPEL, which works by checking event data logs from the services against their custom specification. Two dimensions are defined in this context: *fitness* being the observed behaviour conforming to the specification and *appropriateness* being the resulting evaluation of whether the specification is an appropriate description of how the service should behave. This transitional type approach, as in non-conformance being tested through the expectation of related entities is in a similar vein to how we implement deviation detection pre and post conditions, which are applied to process activities. The transition between process activities can be thought of in this respect, with pre and post conditions being defined to govern how related activities should have been performed and when to constitute a legal transition into the respective activity.

The rest of this paper is comprised of two main sections. In section 2, we present our methodology and describe how our approach can be generically implemented with minimal intrusion into an already existing process. We detail our own definition of non-conformance and how it can be detected through the comparison of a process enactment to its related specification. Section 3 presents a simple implementation in which we applied our methodology to a case study over two distinct phases. The results are then analyzed and discussed before concluding the paper and stating our intention for related future research.

2. Methodology

2.1 Overview

The methodology we describe in this section aims to detect instances of non-conformance in process enactments via a method we call *processspecification enhancement*. This approach is intended to be observational rather than a process support system, which have been criticized in the past for their rigidity and lack of flexibility [11, 12]. The proposed approach is rule based and essentially complements an existing process specification without requiring any modifications to it, and without interfering with the way the process is enacted. We achieve this by defining specific *rule-sets* that stipulate how certain parts of the process should be performed. We then check the validity of these rules by comparing them against process enactment data.

The assumptions we make in this context are that the process enactment data is in such a form that it is possible to use it in rule comparisons and also that the process is broken into specific activities to which we can apply the rule-sets.

2.2 Non-conformance

Non-conformance is defined as any instance where an enacted process has not conformed to its associated specification. In the context of this approach, an instance of non-conformance is any recorded process enactment data that fails to comply with the rules defining the constraints on how the process should or should not have been performed.

As mentioned in the introduction, an instance of non-conformance can be one of two possible types: deviations or inconsistencies. Inconsistencies we define as part of a process enactment that has failed to comply with rules governing either general process constraints as a whole or with the values relating to a specific process activity. A deviation we define as when a process enactment fails to comply with the rules defining a process activities preconditions and post-conditions, representative of the legal transition between activities. Figure 1 depicts this differentiation.

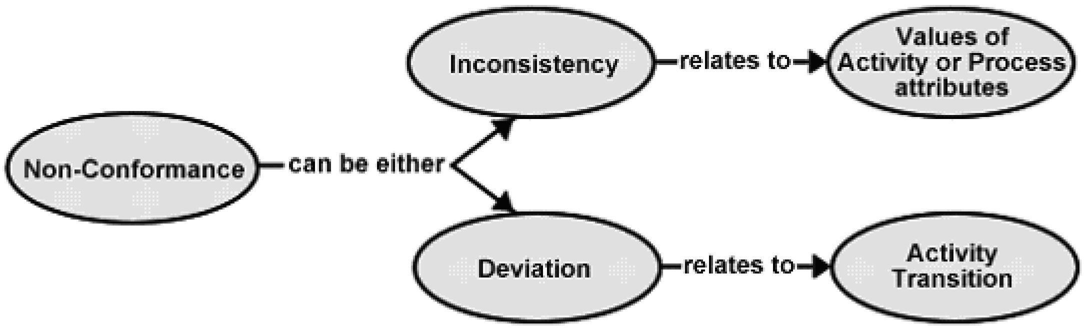


Figure 1. Non-Conformance Type Differentiation

Aside from this basic definition of non-conformance, we extend the concept in order to organise it into an entity capable of delivering as much useful information as possible about an instance of non-conformance once it is detected. In addition to the raw non-conformance data, this also includes information that is extrapolated by comparing the non-conformance instance to other previously detected instances, it's severity in relation to the process and any decision that was made after detection to mitigate any consequences.

These extended attributes include timestamps for when the non-conformance was detected and when it actually occurred, an occurrence count indicating how many of these non-conformance instances have been detected, an importance rating indicating the relative severity of the non-conformance in process and organisational terms and a history log that serves as a record of remedial actions taken post-detection. The full non-conformance definition, and how it relates to the process is illustrated in the concept map provided in figure 2.

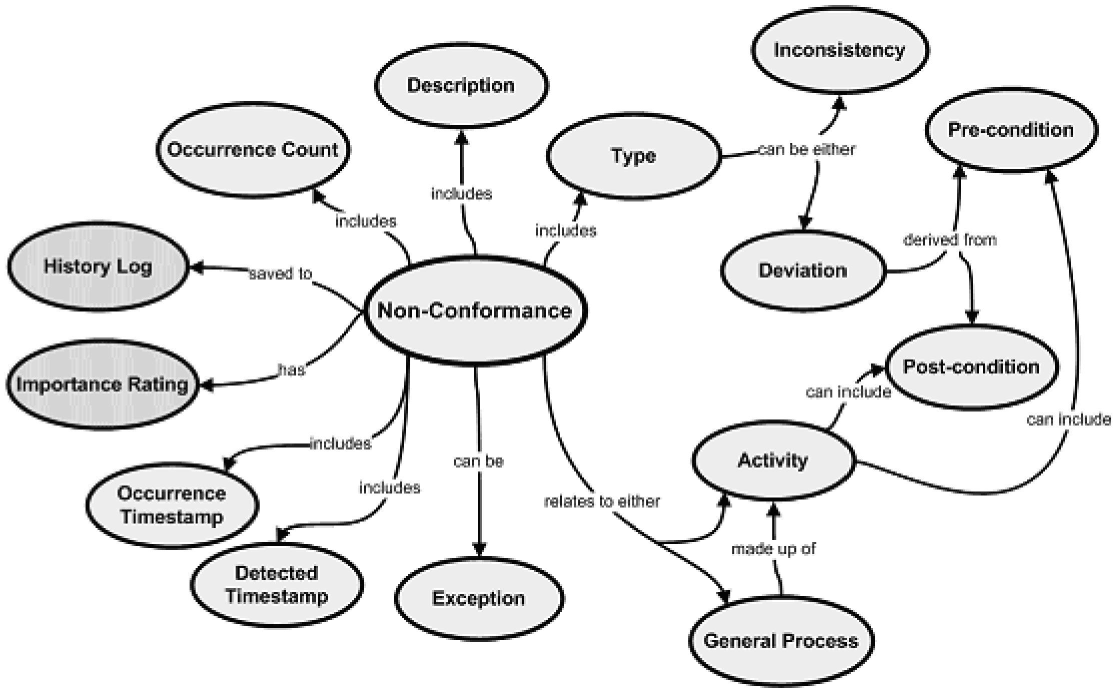


Figure 2. Non-Conformance Concept Map

2.2 Rules

In order to enhance the process specification, we define rules that describe how parts of the process should and should not be legally carried out. Rules are used to check whether process enactment data complies with the specification and will return a Boolean value indicating whether or not the data has passed the comparison check. To illustrate, we use an example from our case study. Suppose there is an existing specification for a process activity called “Fill in survey”, in which a user answers some survey questions. One of the questions is “what is your current character level?”. We can “enhance” this specification by defining a specific non-conformance rule such as:

Example rule:

Character level must be of numeric value

The rule will check to see if the response data submitted by the user is numeric, and if so will return true to indicate that the check passed. If not, then the rule has not been complied with and non-conformance is detected because the false return will cause the rules rule-set to fail (covered in next section). The application and checking of this rule has no bearing on the existing specification for the process activity.

2.3 Rule-sets

Rule-sets are used to group related rules together to give non-conformance instances more meaning and also to make the application of this methodology more scalable and easier to maintain. Rule-sets consist of one or more rules and return a Boolean value in their own right. The Boolean value returned by a rule-set is indicative of the result of the validation of its associated rules.

For each rule-set, every rule associated with it must return true for the rule-set to pass. If one or more rules inside the rule-set return false, the rule-set itself will return false and indicate an instance of detected non-conformance. Rule-sets also have descriptions associated with them, which tell us more about the nature of the non-conformance it is checking for as opposed to solitary rules. We can continue from the previous example on rules to show an entire rule-set for checking the conformance of the specified character level:

Example rule-set:

Rule 1: Level must be numeric

Rule 2: Level must be greater than 0

Rule 3: Level must be less than 80

This approach considers an instance of non-conformance to have been detected when a rule-set returns false. Therefore, rule-sets must return true in all cases in order to avoid non-conformance detection.

2.3.1 Exceptions

Exceptions are a feature of this approach inspired by the work published on "tolerated" non-conformance by Cugola et al. [15]. Exceptions offer the capabilities of checking for non-conformance types that may not necessarily qualify as "non-conformance" in the scope of the process but we still may want to monitor anyway. This is usually a situation that ideally should not occur, but we expect that it might and choose to tolerate it and want to check for it anyway. Exceptions in this approach are treated exactly the same as regular non-conformance instances, only the rule-set that is used to detect them has the "exception" attribute flagged to indicate that these instances are exceptions.

2.3.2 Logic Types

The "logic-type" is an attribute of the rule-set that stipulates how the rules contained within it will be validated. When the rule-set evaluates each of its contained rules, this attribute tells the rule-set how to use the Boolean return value from the rules to determine whether or not the rule-set itself will return true or false.

The reason this attribute was introduced to the rule-set specification is to add greater flexibility in the different ways that the rules and the enactment data can be evaluated. This also helps the rule-sets and therefore the approach become clearer and easier to implement.

The different methods that the rules can be validated are very similar to the way the logic employed by logic gates in digital logic works. At present, the Logic-Type attribute of rule-sets can be set to one of the following four values:

- AND
- NOT
- OR
- XOR

The default logic type for rule-sets if it is not explicitly defined, is the "AND" logic-type. Similar to a digital logic "AND" gate, this logic-type indicates that every rule within the rule-set must return true for the rule-set to return true. If one or more rules return false then the rule-set returns false too, constituting a detected instance of non-conformance. An example rule-set using the "AND" logic type that may be used to constrain a persons age within a process is provided in table 1:

Rule	**Return Value**
Age input is numeric	True
Age input is >= 0	True
Age Input is < 120	True
Rule-Set Return Value:	**True**

Table 1. Rule-set example with an "AND" logic-type

Alternatively, the logic-type attribute can be set to "NOT", which means that all rules must return false in order for the rule-set to return true; "OR", which means that at least one rule within the rule-set must return true; or finally "XOR", which means that one and only one rule within the rule-set must return true in order for the rule-set to return true and non-conformance not to occur.

2.3.3 Remedial Actions

Once an instance of non-conformance has been detected, we provide the opportunity to record the consequential actions that were taken (if any) and provide a user defined score indicating the success of this remedial action. This “score” does not at present have any predefined value constraints, it is simply a mechanism for people to manually record the perceived success level of post-detection decisions. This is catered for via the “history log” of remedial actions, which logs which remedial actions were taken for which detected instances of non-conformance, so that in future there is an easy reference to what works and what does not if there are repeat detections of the same non-conformance types.

The reason this provision is included is so that if and when the same or similar non-conformance instances are detected in the future, there is a record for what was done, if anything and how successful that action was. This affords another avenue using the approach for process improvement. At this stage in non-conformance detection research, consideration for the decision making process after detection has not been addressed in the literature. This represents an important contribution in non-conformance detection research. Even though post-detection decision support is in its research infancy, the introduction of consideration for it in the storage design of non-conformance instances is an important step forward.

2.4 Applying the rule-sets

Zero or more rule-sets can be applied to different parts of the process specification to detect non-conformance, meaning that we can easily check for as much or as little non-conformance as we need to simply by the definition and application of more and new rule sets. We can apply rule-sets to a process in four separate ways:

- General Process
- Activity
- Activity Pre-conditions
- Activity Post-conditions

With respect to activities, we can define general consistency rule-sets for the activity as an entity and also deviation rule-sets that apply to its pre and post conditions. Figure 3 shows the three ways we apply rule-sets to activities.

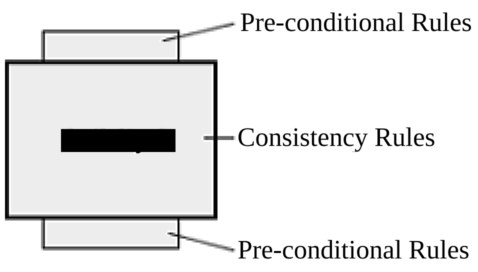


Figure 3. Activity rule-sets

General process rule-sets may also be defined which relate to some of the emergent properties that can only be measured by viewing the process as a whole. These could be rule-sets to govern things such as total activity counts, the absence of any required activities from the process or even just a minimum and/or maximum boundary on the total time duration of the process as a whole.

2.5 Activity sequencing

Since deviations refer to the transition between process activities, we also need to address the way a process may have specified the sequencing of its activities. This regards the conformance of the actual transition between activities rather than a comparison of sequencing patterns such as in [14]. Fortunately, the way this is best covered is through the application of pre and post-conditions governing the legal begin and end conditions of each activity. We can achieve this by simply defining rule-sets for each activity and assigning them as either pre or post-conditions. Instead of defining a situation where activity A has some sort of tag specifying that when it has concluded, activity B should be commenced, it works much better to assign a pre-condition to activity B stipulating that activity A should be successfully completed before activity B should begin.

Park et al. illustrate how pre-requisites can be applied to eLearning activities that govern when each activity can legally begin [8]. The pre-requisite approach stipulates certain conditions that must be complied with before a certain activity can legally commence. This tactic is employed in our own methodology with the use of pre-conditional rule-sets, being rules that govern when the activity can commence without causing non-conformance and also post-condition rule-sets being rules governing when the activity can legally conclude.

2.6. Formalization

As previously stated, we expect that an existing process specification would be apparent in some form, and that the process would be split into activities before we can begin applying this non-conformance methodology to the process specification. This is depicted in figure 4 in the grey area, which represents an existing process specification. Figure 4 also depicts the application of rule-sets to both the process as a whole entity along with its associated activities.

When a rule-set returns false, indicating that an instance of non-conformance has been detected, this is recorded according to the "non-conformance" class related to the rule-set. The rule-set provides general information about the non-conformance such as a description of what was looked for and found, the type and the section of the process it was found in. The detected instances represented by the "non-conformance" class provide more detail about the actual detection characteristics, such as a timestamp or the occurrence count of the detections from the same rule-set. The attributes associated with a detected instance of non-conformance are therefore available through the relationship between the rule-set and the non-conformance classes.

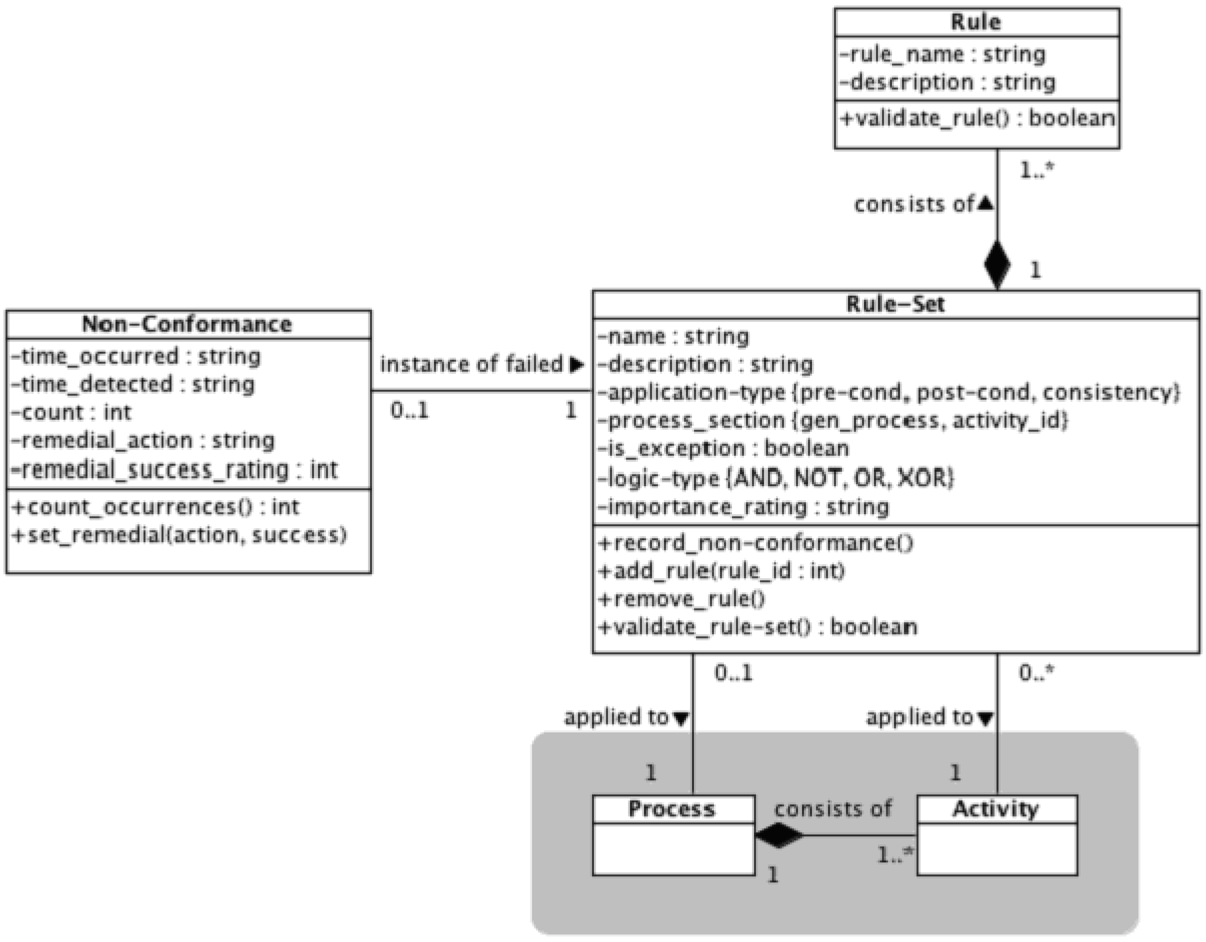

Rule
-rule_name : string
-description : string
+validate_rule() : boolean
1..*
consists of ▲
1
Non-Conformance
-time_occurred : string
-time_detected : string
-count : int
-remedial_action : string
-remedial_success_rating : int
+count_occurrences() : int
+set_remedial(action, success)
instance of failed ▶
0..1
1
Rule-Set
-name : string
-description : string
-application-type {pre-cond, post-cond, consistency}
-process_section {gen_process, activity_id}
-is_exception : boolean
-logic-type {AND, NOT, OR, XOR}
-importance_rating : string
+record_non-conformance()
+add_rule(rule_id : int)
+remove_rule()
+validate_rule-set() : boolean
0..1
0..*
applied to ▼
applied to ▼
1
1
Process
consists of
Activity
1
1..*

Figure 4. UML representation of the methodologies application

3. Case Study

3.1 Overview

To test this methodology we built an online survey conducted in two phases based on the immensely popular online game World of Warcraft. An implementation of this nature provided two important advantages. The first advantage is that due to its online environment, it facilitates easy monitoring and recording of process enactment data. While the actual recording of enactment data is not a problem directly related to this kind of research, it is still a problem that must be addressed in order for any non-conformance detection approach to work. This can be tricky to accomplish and the solution will almost always be specific to the processes operating environment. Since this process was implemented online, the LAMP based environment of the survey enabled us to achieve this with minimal hassle and record exactly how participants went about completing it.

Second, the relatively enormous World of Warcraft community (over 11.5 million subscribers world-wide at the time of this experiment [16]) provides access to steady stream of willing participants through the game's official forums [10]. These people are all available to participate since the survey process was conducted online, provided they are active World of Warcraft players. The large number of participants means that a large amount of data became available with which to properly test the approach over two phases, strengthening the evidence that such an approach can contribute significantly to process improvement.

Survey participants were not aware that the content of their responses were not of direct interest to us as researchers, rather the way that the survey was filled out, or rather how the process of filling in the survey was conducted. In the first implementation phase, the survey was presented to participants to fill out and their responses and behaviour were recorded. The results were then analysed and our non-conformance methodology applied. Using the non-conformance results only from what our approach detected over the first phase, some subtle changes were made to the survey in order to see if and how the survey could be improved using our approach. We then presented the survey again to a new set of World of Warcraft players to complete and again analysed the results.

3.2 Survey Basics

This experiment was conducted in two phases. First, the survey was presented toparticipants as an online survey would normally be. This survey was filled out by 213 different people. The data was collected and analyzed with our non-conformance methodology. Based solely on the findings from the non-conformance detections, a few small improvements were made to the survey presentation and we recollected the data in the second phase, this time from 197 different people. This gave us a dataset from the responses of a total of 410 different people, with enough in both phase one and phase two to adequately analyze the effect of improvements made as a result of the non-conformance analysis.

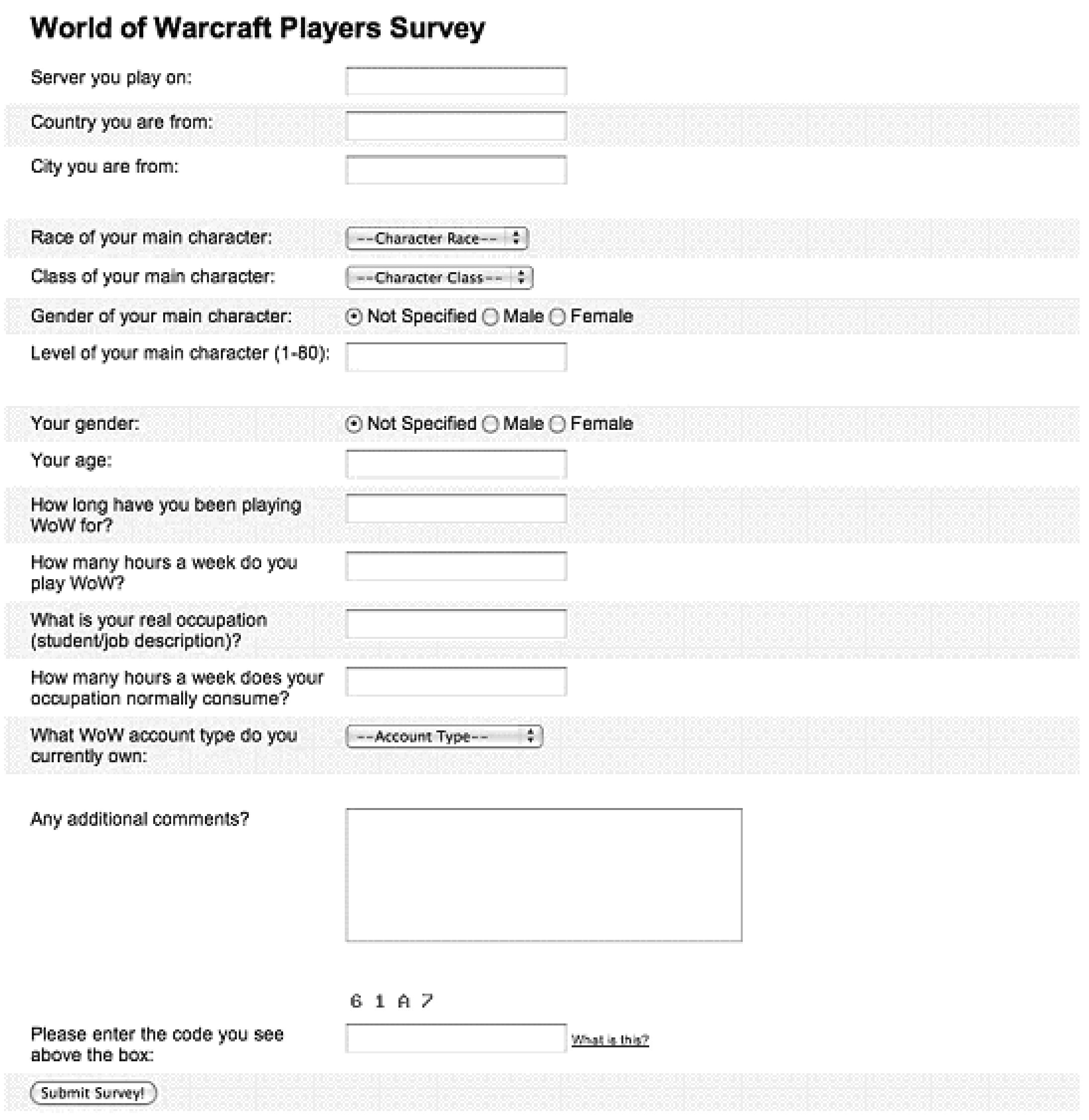
The survey process specification was split into three distinct activities:

- Answer survey questions.
- Validate the security captcha.
- Navigate to the captcha description page.

Rule-sets were defined for each activity and the process as a whole and were applied to the appropriate sections as consistency, pre-conditions and post-conditions as appropriate in PHP code. As each participant filled the survey and completed the process, every single action the participant took was recorded and stored in the MySQL backend database. This includes all failed attempts, timestamps etc.. When a user failed to validate the captcha and was forced to resubmit, the process retained all entered data so that the participant did not have to recomplete the "Answer questions" activity if they failed the "Validate captcha" activity.

A screenshot of the survey is provided in figure 5. A disused version of this survey is also still available at http://wowsurvey.nostin.com for reader convenience. This screenshot is taken from the "phase two" version of the survey.

After phase one of the process was completed, the non-conformance rule-sets were run over the enactment data recorded from all the survey participants. Based on the rule-sets in effect, the resulting non-conformance data was collated and tabulated and is shown in Table 2. This represents which rule-sets were broken, where in the process they were applied and how many violation occurrences were recorded. This was yielded from 213 different people who conducted the survey a total 286 times.


World of Warcraft Players Survey

Server you play on:	
Country you are from:	
City you are from:	
Race of your main character:	--Character Race--
Class of your main character:	--Character Class--
Gender of your main character:	[✓ Not Specified] Male Female
Level of your main character (1-80):	
Your gender:	[✓ Not Specified] Male Female
Your age:	
How long have you been playing WoW for?	
How many hours a week do you play WoW?	
What is your real occupation (student/job description)?	
How many hours a week does your occupation normally consume?	
What WoW account type do you currently own:	--Account Type--
Any additional comments?	

6 1 A 7

Please enter the code you see above the box: What is this?

Submit Survey!

Figure 5. Screenshot of the second phase survey

Rule-Set Broken	Process Area	Type	Occurrences
Account type inconsistent with character level	Answer Questions Activity	Consistency	1
Age	Answer Questions Activity	Consistency	25
Character Level	Answer Questions Activity	Consistency	20
Race and Class mismatch	Answer Questions Activity	Consistency	11
Captcha code and user code mismatch	Validate Captcha Activity	Post-condition	81
Captcha fail more than once	Overall Process	Consistency	16
Overall time duration	Overall Process	Consistency	49
Validate captcha activity must not be complete	Navigate to “What is captcha” Activity	Pre-condition	2
Questions Complete	Answer Questions Activity	Post-condition	5

Table 2. Phase One Non-conformance Instances

Based only on this non-conformance data, there are a few glaring areas in the process that require addressing. Most obvious here is that participants are having an overly difficult time validating the captcha code. It was failed a total of 81 times in 286 attempts survey wide. Further analysis of the non-conformance data showed that 16 participants violated this rule set more than once, meaning that 7.4% of participants triggered non-conformance from this section of the process on more than one occasion.

Less obviously, the relatively high number of "Character Level" rule-set violations is alarming. It can be assumed that registered players should know very well the numeric range of possible levels (1 - 80), so 20 violations is relatively high and cause for concern.

- From this analysis of the first phase, the following subtle alterations were made to the process with the intention of improving it:
- Reduction of the number of captcha code characters from 6 to 4.
- Elimination of all lowercase characters, leaving only uppercase and numeric.
- Elimination of the O and the 0 from the list of possible code characters – they might look too similar for users to distinguish.
- A capon the character level text field of 2 characters

Phase two was then conducted, this time with 197 participants completing the process a total of 217 times. Table 3 shows the results.

Rule-Set Broken	Process Area	Type	Occurrences
Race/Account mismatch	Answer Questions Activity	Consistency	1
Account type inconsistent with character level	Answer Questions Activity	Consistency	2
Age	Answer Questions Activity	Consistency	7
Character Level	Answer Questions Activity	Consistency	8
Race and Class mismatch	Answer Questions Activity	Consistency	3
Captcha code and user code mismatch	Validate Captcha Activity	Post-condition	22
Captcha fail more than once	Overall Process	Consistency	1
Overall time duration	Overall Process	Consistency	24

Table 3. Phase Two Non-conformance Instances

As can be seen from the second phase data, the figures are noticeably better where the changes were made. First, the number of character level rule-set violations have dropped from 20 to 8. This has roughly halved the failure rate from 7% to 3.7%.

The biggest improvement was in the captcha code violation non-conformance violation rate. 22 failures out of 217 represents a drop from a failure rate of 28.3% to 10.1%. Furthermore, there was only a single instance of a participant triggering the captcha violation more than once, which is a drop from 7.4% in phase one to 0.5% in phase two.

These improvements in the data can be directly linked to the corrective actions made as a result of the phase one non-conformance detections.

3.3 Evaluation

From all of the non-conformance instances detected in the first phase, the most significant was the captcha code failures. The primary purpose of a captcha on a website is to differentiate between human and non-human users [9] thus automatically removing any entries deemed to have been submitted by bots or crawlers. It seemed that people were having a lot of trouble filling in the code correctly, because the non-conformance detected related to records that were otherwise indicative of a genuine human response. By taking the corrective measures described in section 3.2 we were able to reduce the number of captcha failure non-conformance instances from 7.4% of participants to 0.5%.

Of course, the measures taken were in speculation at the time. The data yielded from the non-conformance detections did not tell us to reduce the number of captcha characters or to remove the 'O's and '0's. What it did was highlight that there was a problem with that section of the process, and left it up to the administrator's expertise to decide how best to rectify it. Therein lies one of the inherent values in non-conformance detection, it highlights that there is an issue and where that issue is so that we are aware of it and can take steps to address it.

We were pleasantly surprised to discover that a lot of the value in non-conformance detection is that it tends to point you to look in a direction you may not have otherwise, prompting you to spot issues with the process specification itself and not necessarily from the enactments.

Another useful feature was noticed upon discovering that when a record has a high number of non-conformance instances detected, it usually meant that a participant was deliberately messing with our survey. If we actually cared about collating the survey question data for research purposes, this would aid greatly in weeding out the invalid responses without having to manually iterate such a large dataset.

4. Conclusion and Future Work

The work presented in this research yielded some interesting results. We believe that the application of this methodology achieves the research goal of being simple and easy to implement however this is going to be dependent on the process in question and the accessibility of its enactment data. The effectiveness of our proposed approach is also going to be heavily dependent upon the competence of the administrator to identify and define appropriate non-conformance rules and rule-sets.

Since there has been no research conducted so far on supporting remedial decision-making after non-conformance has been detected, this appears to be a promising avenue in which to conduct further research. The concept is introduced briefly in this paper, but there is no robust methodology on offer to help process administrators make informed decisions once non-conformance has been detected based on what would be best for process improvement and how significant the detected non-conformance instance was in relation to both the process and the organisation.

In this vein, we believe there is still an opportunity to further develop this approach through the forming of more generic rules which could be developed ready for implementation without the need of the process administrator to identify and specify them. This is not to remove the flexibility of having administrators define their own rules, but with experience the more common rules and rule-sets could be more easily identified and "templates" of sorts could be developed for their easy integration. These could be pre-defined rule-sets to govern things like a generic implementation of activity time duration boundary values or even domain specific rule-sets that could be relevant to a particular process types environment or domain.

The consistent nature of resource usage in processes also lends itself well to this idea, and our existing research on resource based non-conformance detection [13] is a good candidate for exploring this in our future research.

References

[1] Rezgui, Y., Marir, F., Cooper, J., Yip, G., Brandon, P,. (1997) A Case-Based Approach to Construction Process Activity Specification, *International Conference on Intelligent Information Systems (IIS)*, 8-10 Dec, p. 293 – 297.

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

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

[4] Huo, M., Zhang, H., Jeffery, R. (2006). An Exploratory Study of Process Enactment as Input to Software Process Improvement, *In*: International Conference on Software Engineering, Shanghai, China.

[5] Cîmpan, S., Oquendo, F.(2000). Dealing with software process deviations using fuzzy logic based monitoring**,** *ACM SIGAPP Applied Computing Review***,** 8 (2) ACM Press, December.

[6] Kabbaj, M., Lbath, R., Coulette Bernard, B. (2007). A Deviation-tolerant Approach to Software Process Evolution, *In*: Ninth international workshop on Principles of software evolution (IWPSE'07), Dubrovnik, Croatia, September 2007.

[7] van der Aalst, W.M.P., Dumas, M., Ouyang, C., Rozinat, A., Verbeek, E (2008). Conformance checking of service behavior, *Transactions on Internet Technology (TOIT),* 8 (3) ACM.

[8] Park, N., Kim, H., Kim, K., Park, H., Chun, J., Hwang, Y. An eLearning Activity Control Model for SCORM's Sequencing Prerequisites, *In*: Fourth International Conference on Semantics, Knowledge and Grid, IEEE, 3-5 Dec. p. 322 – 329.
[9] Official CAPTCHA site, http://www.captcha.net.
[10] World of Warcraft official game forums, http://forums.worldofwarcraft.com/index.html
[11] Ramage, M. (1994). Engineering a smooth flow? The links between workflow and business process reengineering, MSc thesis, University of Sussex, England.
[12] Aversano, L., Canfora, G. (2002). Introducing eservices in business process models, *In*: Proceedings of the 14th International Conference on Software engineering and knowledge engineering SEKE '02, ACM Press, July.
[13] Thompson, S., Torabi, T. (2008). Towards Formalizing Resource Based Non-Conformance in Business Processes, *In*: 19th Australian Software Engineering Conference ASWEC, Perth, Australia, March.
[14] Hamid, R., Johnson, A., Batta, S., Bobick, A., Isbell, C., Coleman, G. (2005). Detection and explanation of anomalous activities: representing activities as bags of event n-grams, *In*: Conference on Computer Vision and Pattern Recognition, IEEE, 1, 20-25 June, p. 1031 – 1038.
[15] Cugola, G., Di Nitto, E., Ghezzi, C., Mantione, M (1995). How to deal with deviations during process model enactment, *In*: Proceedings of the 17th international conference on Software engineering, ACM Press, April.
[16] Blizzard Entertainment Press Release (2008). WORLD OF WARCRAFT® SUBSCRIBER BASE REACHES 11.5 MILLION WORLDWIDE, http://www.blizzard.com/us/press/081121.html 23 December 2008.