A METHODOLOGY FOR INFORMATION QUALITY MANAGEMENT IN SELF-HEALING WEB SERVICES
(Completed paper)

Cinzia Cappiello
Barbara Pernici
Politecnico di Milano, Milano, Italy
{cappiell, pernici}@elet.polimi.it

Abstract: Self-healing systems have attracted vast attention by the research community. Self-healing Web services have the ability to perceive that they are not operating correctly and, without human intervention, make the necessary adjustments to restore itself to normal operation. Information quality is a fundamental aspect of Web services since wrong or out-of-date information can cause the failure of the service. Measuring and improving information quality is a complex task. Several methodologies have been developed in the last years to provide a support for the design and implementation of assessment and improvement techniques in order to guarantee high data quality level. The existing approaches focus on the system design phases in which enterprise context is analyzed, quality dimensions are selected, assessment and improvement techniques and tools are modeled. In this paper, a methodology is, instead, proposed to be a support to self-healing environments and to solve run-time data quality problems. The analysis of the business processes and context in the design phase allows identifying critical points in the business tasks in which information quality might be worsened. In these points, information quality blocks have to be inserted in order to continuously monitor the information flows. Through suitable checks, failures due to information quality problems can be detected. Furthermore, failures and warnings in service execution may depend on a wide variety of causes. Along the causes, the methodology also produces a list of the suitable recovery actions for a timely intervention and quality improvement. In particular, the focus of the paper is on data quality problems introduced by a shared use of data in different processes or by periodically synchronized data sources in process based systems using Web services. The methodology is explained by means of a running example.

Key Words: Information quality, Self-healing Web service, Warning Management.

1. INTRODUCTION
The term 'Web services' represents the overall concept of technologies and programming standards to enable independent software applications to interact with each other over existing Internet protocols. Nowadays, Web services are widely studied both in the academic and business sectors since they have potentially a lot of advantages in terms of interoperability, integration, scalability, and reuse. The implications of this new technology are enormous and widespread: a lot of business applications such as ERP and CRM platforms, B2B and B2C businesses, may be transformed by the emergence of Web services. Actually, even if various efforts have been recently done in Service Oriented Computing, the adoption of Web services still remains limited. This is because Web services standards features such as transactions are currently nonexistent or still in their infancy compared to more mature distributed computing open standards such as CORBA. In fact, Web services may suffer from poor performance compared to other distributed computing approaches. Furthermore, if we consider public available Web service registries, as UDDI, it is easy to find Web services no longer available or Web services having a description which does not correspond to the currently provided Web services, due to new versions or modifications of functionality of the Web services. In summary, considering both functional and non functional requirements (related to the Quality of Service), it is possible to enumerate many possible causes of failures that obstacle the diffusion and use of this new technology.
In order to overcome these issues, the study and the development of solutions for the creation of self-healing Web Services are increasing. Self-healing Web services are able to detect anomalous situations, which may manifest as the inability to provide a service or to fulfill functional or non-functional requirements, and to recover from these situations, e.g., by rearranging or reconfiguring the network of services. It is possible to identify different types of faults, due to architectural or application problems. In this paper we focus on faults related to the information that Web services exchange among each other. In fact, exchanging wrong or out-of-date data can cause the failure of the application both in case that information is the real output of the service or that it is only used in the control flow.

Information quality aspects are still scarcely considered in the literature about self-healing systems, even if recently, they have attracted vast attention by the research community. An overview of self-healing concepts can be found in [9], where the main elements of the self-healing system problem space are presented, that is the fault model, the system response, the system completeness, and the design context. Examples of self-healing architectures is provided in [14]. The architecture presented in [14] is focused on service level agreement management in a Service Oriented Architecture. The goal is to re-optimize the provisioning infrastructure as a consequence of QoS violations. If an abnormal event is detected by the monitoring infrastructure, the observed evidence is loaded into the causal network, and the reasoning engine provides as a result the possible causes of QoS violations. When a fault is identified, the recovery action specified in the service policies is executed. In this work the quality of information as part of the QoS is not considered.

Data are instead considered in [1][2][7], in which authors propose a model based approach to implement diagnosis functionalities in Web services self-healing environments. The goal is to correlate input and output parameters and to determine if an activity carries out a computation that may fail and produce erroneous outputs. The proposed Web service model considers, together with other variables, the information flow of a single process. In this work the abnormal value in output of an activity is not related with data quality dimensions and relations among different sources and different processes are not investigated and modelled.

In our perspective, it is possible to state that information (or data) quality is a considerable aspect of the quality of service and in the last few years its importance is constantly growing. In general, in order to assure high data quality level in the offered Web services, organizations should continuously check the quality of the owned and exchanged data. Organizations should implement a complete data quality management program to support the four classical phases of data quality management: data quality definition, measurement, analysis, and improvement. In the data quality definition phase, the set of data quality dimensions to measure is identified. Then, after the measurement process, the results of the evaluation are analyzed and poor quality data situations are detected and sent as input to the improvement phase. In the improvement phase, the collection of poor quality data cases is deeply investigated and an improvement action is suggested. The four phases are iterated in this order over time to guarantee the continuous control of data values. In the literature different methodologies have been proposed in order to provide guidelines, and thus support data quality management. Most famous complete and general methodologies are the methodology developed at the MIT (Massachusetts Institute of Technology), called the Total Data Quality Management (TDQM) [15] and the TIQM (Total Information Quality Management) methodology [8]. These data quality programs in general require a considerable personalization effort before application. In fact, a big limit in the application of these methodologies is that the authors provide a series of guidelines without providing tools or specific algorithms that support users in the assessment, monitoring, and improvement phases.

However, the improvement activities in case of poor quality data are always defined as results of a periodical data quality analysis. In the literature, methodologies that manage the run time errors detection and recovery are missing. The methodology proposed in this paper is a first attempt to provide a
methodological approach at the run time error detection and correction management. In fact, detection of errors in the run-time phase of the process is supported and suitable improvement actions are enforced. The existing approaches focus on the system design phases in which enterprise context is analyzed, quality dimensions are selected, assessment and improvement techniques and tools are modelled. In this paper, a methodology is instead proposed to be a support to self-healing environments and to solve run-time data quality problems. In practice, the methodology provides service execution mechanisms that guarantee diagnosability and repairability of run-time failures. The diagnosability is guaranteed by the analysis of the business processes and context in the design phase and the identification of critical points in the business tasks in which information quality might be worsened.

In fact, data tracking methods are required in order to determine the exact stage or steps in information process where the causes of data quality decreasing occur [3]. In the literature, it is possible to find several methods that allow the representation of a process and the associated information flow in order to detect errors and facilitate data quality improvement. An example is the IP-MAP methodology [12]. Once the information product has been selected and its quality specifications are well understood, in this phase it is possible to construct the Information Product Map (IP-MAP). The information product map graphically describes the process by which the information product is manufactured. There are different types of construct blocks that form the IP-MAP but the most important innovative feature is the introduction of “data quality block” that is used to represent the checks for data quality on those data items that are essential in producing a “defect-free” information product. The representation of the process by using IP-MAP enables the detection of different anomalies in data flows. Thus, by using suitable checks, failures due to information quality problems can be detected. Furthermore, failures and warnings in service execution may depend on a wide variety of causes. Along the causes, the methodology also produces a list of the suitable recovery actions for a timely intervention and quality improvement.

This paper is organized as follows. Section 2 describes the methodology from a general point of view, by illustrating the phases that compose it and focusing on innovative aspects as regards the literature. Section 3 focuses on the Warning Management module and the model that is used to detect errors. Section 4 provides the description of the tool that has been implemented to support the methodology.

2. THE METHODOLOGY FOR QUALITY MANAGEMENT IN SELF-HEALING ENVIRONMENTS

The methodology is a complete methodology for Information Quality management and improvement and it is composed of eight steps (see Figure 1) [6].

![Figure 1- Data quality management architecture for self-healing environments](image-url)
Considering both the classical four phases of the data quality management program discussed in Section 1 and modules represented in Figure 1, it is possible to identify the following mappings: the Environment Analysis, Resource Management, and Quality Requirements Definition are all referable to the Definition phase; Quality Measurement and Analysis & Monitoring modules have the same functions of the classical approach, while Improvement & Real Time Recovery and the Strategy Correction modules are related to the Improvement phase.

Let us introduce a further distinction by identifying two phases: data quality definition phase and data quality evaluation phase. This classification is useful in the approach presented in this paper in order to distinguish the modules that are used at design time and modules that are used at run-time. The following sections provide details about each phase. Note that the Warning Management is an innovative contribution of the methodology for the real-time warning management and it is described in Section 3.

2.1 Data Quality Definition phase
As stated in Section 1, a data quality program should be based on a thorough knowledge of business context and organizational data and processes. In the methodology, this knowledge is acquired in the IQ Environment Analysis and Resource Management steps. The former step is composed of different tasks in which data sources, processes, and stakeholders of the organizations are identified, and strategic issues are considered. In fact, in this phase, there is the analysis of the feasibility of the implementation of the data quality management program and thus the definition of the management plan together with the identification of potential critical issues, risks critical success factors, and performance key indexes. The management plan is the input for the Resource Management step in which there is the definition of all the resources that have to be involved in the data quality program. Thus, starting from the business processes that have to be monitored, it is possible to identify in this phase the data sources, the applications and the human resources that are related to each business process. The Data Quality Requirements Definition step is the most important part of the Definition phase since it is based on the concept of the user perspective. The methodology proposed in this paper is different from the other methodologies proposed in the literature since it considers the different information stakeholders and models their preferences and needs. Details of this module can be found in [6].

2.2 Data Quality Evaluation phase
The Data Quality Evaluation phase includes the Quality Measurement, Analysis & Monitoring, Improvement & Real Time Recovery, Strategy Correction and Warning Management steps. Our approach here is quite similar to the existing methodologies apart from the Warning Management step that is a novelty in data quality management. In the Quality Measurement module, for each quality dimension, a measurement algorithm has to be identified. The results of the evaluation are sent as input to the Analysis & Monitoring step that compares the values associated with each quality dimension to the quality requirements. If the requirements are not satisfied, the system is in charge to find out the causes of the poor quality and along these causes, suitable improvement actions have to be identified. In order to achieve the fixed data quality goals, data-oriented and process-oriented techniques are used in the improvement phase. The most straightforward solution suggests the adoption of data-oriented inspection and rework techniques, such as data bashing or data cleaning, to solve problems related to data accuracy and data consistency. A fundamental limitation of these techniques is that they do not prevent future errors, so they are considered appropriate only when data are not frequently modified [11]. On the other hand, a more frequent use of data bashing and data cleaning algorithms involves high costs that can be difficult to justify. To overcome these issues, several experts recommend the use of process-oriented methods. These methods allow the identification of the causes of data errors and their permanent
elimination through changes in data access and update activities. These methods are more appropriate when data are frequently created and modified.

In critical situations, improvement actions also imply modifications at a strategic level, that is, changes in data quality planning system and requirements. These structural changes are planned and performed in the **Strategy Correction** step.

### 3. The Warning Management Module

The Warning Management phase represents an innovative contribution of the proposed methodology. In fact, no other methodologies have proposed an interactive analysis of warning and failure message to assess and improve data quality. Essentially, the warning management represents an internal system for real-time data and processes monitoring. The architecture presented in Section 3.1 has been deeply described in [6] while the data error detection and recovery model illustrated in Section 3.2 is the main contribution of this paper. This model allows the identification of data quality problems introduced by a shared use of data in different processes or by periodically synchronized data sources in process based systems using Web services.

#### 3.1 The Warning Management Module architecture

The Warning phase is managed by different components as represented in Figure 2.

![The Warning Management Module](image)

The **Warning Message Generator** is the core module of the architecture and it works on the basis of the messages sent by a **Diagnoser** and the **Internal/External Feedback** modules. These modules are in charge to monitor the system, detect all the anomalies that occur in data management, and identify sources of quality problems. In details, the **Diagnoser** represents the most important component of the Warning Management system. It identifies and manages fault and warning events by monitoring the internal system and external context. After the warning identification, **Diagnoser** sends the information to the **Message Generator** that produces the warning message that is processed by the **Warning Analyzer**. Three different monitoring submodules supports the **Diagnoser** in faults detection: (i) **System Discrepancy Analyzer** detects all the discrepancies that occur between internal data and external data that are contained in external sources (e.g. Web data, suppliers' data; (ii) **System Inconsistency Analyzer** focuses on quality dimensions values. In particular, it identifies inconsistencies between actual data quality values and
requirements; (iii) **System Monitoring** monitors the system and its behaviour in order to detect anomalies in data management such as outlier detection or peak values in data accesses. These anomalies have to be analyzed since they may be the indicators for data quality problems.

Data quality problems can be also identified on the basis of received feedbacks. Feedbacks can be classified as (i) **External feedbacks** that are generated by external applications or sources that interact or cooperate with the organization platform; (ii) **Internal feedbacks** that are generated by the internal system; (iii) **Subjective feedbacks** that are based on surveys of the stakeholders' satisfaction. The **Internal/External Feedback** module receives the different types of feedbacks and compares them with the interested parties requirements and the required performance indicators in order to identify the significance of the problem. On the basis of this analysis the **Internal/External Feedback** module selects the feedbacks that are the indicators of significant data quality problems. Once the fault has been detected, a warning message is created, stored in the **Warning Log Database**, and sent to the **Warning Analyzer**. This module, on the basis of the rules contained in the **Warning/Recovery Database**, identifies the most suitable recovery action for the detected problem. In fact, the methodology defines at priori rules for problem solving and recovery selection in order make the resolution of data quality issues easier and automated. Thus, **Warning/Recovery Database** contains all the associations between possible poor data quality situations and related recovery actions. Associations are based on considered data, data quality dimensions, access frequencies and so on. For example, if the problem regards the accuracy of a data value that is scarcely accessed, the system will only suggest the immediate correction of the problem; while if data value is frequently accessed, besides the correction, a process-oriented method is required. Finally, **Real Time Recovery** module applies the appropriate recovery actions and monitors their impact to the system during implementation and execution.

Note that each time that a warning message arrives and recovery actions are performed, the system updates data quality values and traditional monitoring operations are performed.

### 3.2 The Data Error Detection and Recovery Model

Data error detection in run-time phase needs for a thorough knowledge of the overall process that characterizes the considered Web service and the quality of the data stored and used in the organization. In fact, the analysis of the business processes and context allows identifying critical points in the business tasks in which information quality might be improved. In these points, information quality has to be assessed in order to continuously monitor the information flows and notice if its level decreases under the acceptable limits defined as non functional requirements within the Web service. In fact, run time recovery actions in data quality require the identification of the causes of data errors and their permanent elimination through an observation of the whole process in which data are involved.

The data error detection and recovery model presented in this section is based on the IP-MAP model [12] since the approach suggests to insert “data quality blocks” that are used to represent the checks for data quality that are essential in producing a “defect-free” information product. In the considered context, data quality blocks are used to evaluate the quality of the information exchanged among activities involved in the business process. Therefore, a list of the data quality checks that are being performed on the specified component data items is associated with each block.

In fact, note that the model assumes that we can see a Web service as a process $p_i$ whose basic constitutes are activities $a_{ik}$ [7]. For each activity, input and output information flows have to be observed. In [7] the data flow variables represent the correctness status of a piece of data at a given point of the workflows execution. They can be associated with two possible values: “ok” and “abnormal values”. With apposite checks on data, we extend the annotation to four values: “ok”, “incomplete”, “inaccurate”, and “out-of-date”. In this way, we are also able to establish the exact nature of the abnormality. Note that the last three values are not exclusive because incomplete data are also inaccurate and incomplete and inaccurate values...
can be also out-of-date. The model could be extended to a larger set of quality dimensions. So far, we have only used accuracy, completeness, and timeliness dimensions but they constitute a minimal set of data quality dimensions that provides sufficient information about the suitability of data along the process in which they are involved. Accuracy and completeness assess data along their numerical extension and correctness [11] [13]. Timeliness evaluates the validity of data along time [4].

In this paper, we start from the consideration that it is not sufficient to consider only input and output information flows, but it is necessary also consider data that are used by the generic activity \( a_{ik} \) but do not derive from previous activities executed in the process. We refer to external data identifying all data that belong to this category. For example, let us consider the activity \( a_{12} \) that is the second activity in the process \( p_1 \) and it is in charge to identify a specific product in a warehouse and to delivery it to a shipper. \( a_{12} \) receives the product code and by accessing local databases, it is able to retrieve its position (e.g. shelf) in the warehouse. The product position is an example of external data.

In general, it is necessary to model the Web service execution and the role of data in it considering both the information flow associated with the process and the data management structure.

According to Figure 3, the output data can be considered as:

\[
\text{Intdata}_{ik} = a_{ik} (\text{Intdata}_{i(k-1)}, \text{Extdata}_{ikj})
\]

where \( a_{ik} \) is the k-th activity of the i-th process that is responsible for the data manipulation; \( \text{Intdata}_{i(k-1)} \) are the output data of the (k-1)-th activity; \( \text{Extdata}_{ikj} \) are the data extracted from the j-th source and used by the k-th activity. Note that the generic j-th source can be a database or a process which information flow is independent from the one related to process \( p_i \).

In order to detect data quality errors, data will be associated with an annotation that is constituted by three variables \(<\text{acc}, \text{comp}, \text{time}>\) that correspond respectively to the accuracy, completeness, and timeliness dimensions. In the model, we assign to each variable a value equal to “1” to indicate a high quality level along the specific dimension and “0” otherwise. The “ok” value as indicator of high quality on all the considered dimensions is expressed with the triple \(<1,1,1>\).

According with this model an error in the output data can be consequence of:
- an error generated by the activities that precede the analyzed one;
- an error generated by the analyzed activity. This type of error can be classified as self-generated error.

A self-generated error can be caused by functional error in the activity execution flow or by low quality external data.

Self-generated errors are caused by functional problems in the execution of the activity that can be related to issues in data structure or external processes or data. An important aspect that has to be considered in data structure is data redundancy: the same data are contained in different databases that are used by different processes possibly in different organizations. In this case, data have to be periodically realigned. Each source is either directly updated by a service, or periodically refreshed from the other sources. In the period between two realignments data can be incorrect due to their “out-of-dateness”. In order to identify this type of situation, the system has to modelled in term of influences among databases. Each \( \text{Extdata}_{ikj} \)
may be related to other activities if the j-th source ($ds_j$) overlaps with other in the system. It is necessary to store for each $ds_j$ the different overlaps. We suppose to have a matrix $ODS$ in which if the single element $ods_{jz}$ is not null the two sources $ds_j$ and $ds_z$ may overlap, that is:

$$ods_{jz} = od_j \cap od_z \neq \emptyset$$

As shown above, data sharing across operational databases raise the need for periodic data alignment. To be as general as possible, we consider realignment parameters for each pair of databases: each pair of operational databases $ds_j$ and $ds_z$ is aligned with a refresh period $rt_{jz}$, which can be seen as the time interval before the data contained in $ds_j$ are updated with data created or modified in $ds_z$. Refresh periods $rt_{jz}$ can be represented in a matrix $RT$. An element of $RT$ is not null if corresponding $ods_{jz} \neq \emptyset$ and if a data unit, which belongs to $ods_{jz}$ is changed in $ds_z$.

Note that the “out-of-dateness” caused by missed or late source alignments has also effects on accuracy and completeness dimensions [5]. In fact, a data unit that belongs to $ods_{jz}$ and is updated or created in $ds_z$ is respectively wrong or missing in $ds_j$ till the next realignment.

Redundancy can also cause accuracy problems due to error execution in activities that not belong to the same process of $a_{ik}$. This could happen if we have to activities $a_{ik}$ and $a_{wx}$ that have external data that derive from the same source (i.e. $Extdata_{ikj}$, $Extdata_{wxj}$) or that derive from overlapping sources (i.e. $Extdata_{ikj}$, $Extdata_{wxz}$ and $ods_{jz} \neq \emptyset$). In order to complete the description of the data structure of the system useful for the recovery action, a matrix that clarifies which activities access which sources is needed. We define a matrix $AS$ in which if the single element $as_{ikj}$ is not null the activity $a_{ik}$ accesses the source $ds_j$. In the matrix is also clarified the type of access by using the CRUD acronym approach.

![Figure 4 - Example of AS (Activity -Source) matrix](image)

In this way, it is possible to identify the causes of detected error not only by identifying anomalous activities in the same process but also in different processes.

In summary, along the presented data quality dimensions and the activity model, it is possible to classify faults along two categories: value mismatch and missing data.

The value mismatch can be derived by:
- Typos: e.g., Jhon instead of John;
- Different format: e.g., date expressed as dd/mm/yyyy instead of mm/dd/yyyy;
- Conflict in data values: e.g., Residence city: London Country: Italy;
- Delay in update operations.

Typos are related to data accuracy while conflicts in data values are related to data consistency. A case of different format can be related to both accuracy, because the value is not a right representation of real-world value, and to representation consistency. Delay in update operations between two databases that contain the same values can be related to both timeliness, since the out-of-date value is not valid anymore, and accuracy since that the out-of-date value is an incorrect representation of the real world value.
Missing data can be caused by value unavailability or by a delay in update operations. The former is related to the completeness dimension. Delay in update operations between two databases that contain the same values can be related to the timeliness dimension but in this case the update operation would create new tuples in a database.

All the variables defined in the proposed model are stored in a process history file which analysis can support (i) the adoption of process-oriented methods and (ii) the identification of the part of the process which has to be modified in order to eliminate the causes of poor quality.

In general, error recovery can be performed using different data and process oriented methods. Data-oriented methods will include: (i) Data cleaning by manual identification: comparison between value stored in the database and value in the real world; (ii) Data bashing (or Multiple sources identification): comparison of the values stored in different databases; (iii) Cleaning using Data edits: automatic procedures that verify that data representation satisfies specific requirements. Process oriented methods modify the structure of the process to which the analyzed activity belongs or is related. For example, in case of out-of-dateness of the values due to delays in realignment a possible recovery action is to force the realignment in order to get the right value and to redefine the realignment frequency in order to decrease the probability of out-of-dateness of the different values.

4. AN EXAMPLE: BOOKSHOP WEB SERVICE

In this chapter, an example of cooperative business process is described in order to explain how the methodology works. Figure 5 describes the architecture underlying a sample process composed by Web services managed by different organizations BookShop WS, Publisher WS, Warehouse WS and Shipping WS and that aims at supporting a customer in buying a book. The customer asks for a book invoking the BookShop WS, and submitting the title of the book.

![Figure 5 - The Bookshop Web Service](image)

We suppose the bookshop owns a set of books catalogs obtained by the associated publishers. So, after receiving the customer request, the BookShop WS retrieves from these catalogs the most suitable book edition with respect to the title, and sends the order request invoking the Web service of the selected publisher, sending order details containing the ISBN of the book, and the set of customer information relevant for the Publisher WS (e.g., the user's address). In the same way, once the order from the BookShop WS arrives, the Publisher WS identifies the warehouse where the book is available and then selects the one (i.e., Warehouse WS) closest to the customer. Each warehouse usually relies on a set of shippers and, once an order arrives, the most suitable one (i.e., Shipper WS) is selected according to the final destination. Notice that, once the book is received by the customer, the shipper sends a notification...
directly to the publisher in which the reference to the concluded order is included. After receiving the notification, the bookshop sends the final receipt to the customer (see Figure 6).

In the proposed example, the goal of the set of interacting services is to send the right book to the users according to their preferences. A number of faults may occur during the execution of the process illustrated above, which may cause the wrong book to be delivered [1], a book to be indefinitely reserved for a user who is not going to buy it, an order to be indefinitely delayed, and one or more of the involved services to be indefinitely running waiting for a reply.

Faults related to the accuracy dimension include the mismatch between user request and available books due for example to data incorrectness since bookshop has publishers’ catalogs with typos. In this case the book can be never retrieved. Accuracy problems also occur if user inserts a wrong information; e.g. the user types a wrong address and the book can be never shipped.

Books could not be retrieved for completeness problems as well. The book could be not completely described in the publisher catalogs or there could be a database misalignment between Bookshop and Publisher (e.g. publisher does not send catalog updates). In both cases the user requirements might be unfulfilled and book search might fail. The misalignment between Bookshop and Publisher databases is a critical issue in terms of out-of-date information too. For example, if the item Cost is not updated in the Bookshop database, the system can assign to the user a book that corresponds to a higher cost in the publisher database; in this case user requirements might be unfulfilled since the user will receive a book that does not satisfy his/her requirements.

Thus, with the model proposed in Section 3, it is possible to identify the activities related with the activity in which the data quality error is detected. Activities can belong both to the same and to different process. Let us consider an example in which two different orders of the same book are received. The system activates two different processes $p_1$ and $p_2$. We assume that the same publisher receives the request and check for availability for the two consequent book requests in the same warehouse. A misalignment between warehouse and publisher databases can cause the failure of a request. In fact, if the warehouse owns a single copy of the book, the book is not available after the first request, but if the publisher database is not updated, publisher continues to see the book available and sends the other order that will fail.
Figure 7- Activities analysis

Figure 7 shows the situation above described. The output of $a_{29}$ is not accurate since the activity does not generate the expected result, i.e. the position of the book on the shelf. Local evaluations reveal that all the internal and external data are characterized by high quality but looking at the AS matrix, it is possible to observe that two activities are accessing to the same source from different processes. Then, referring to the ODS matrix, it is possible to identify the existence of an overlap between $ds_8$ and $ds_9$ and it is possible to infer that this overlap can be the cause of the occurred fault.

5. CONCLUSIONS AND FUTURE WORK

The paper proposes an approach for data quality management in self-healing Web services. The innovative contribution of the proposed methodology is the model for error detection and recovery in the run-time phase. Data quality errors related to accuracy, completeness, and timeliness dimensions are so far managed. In particular, the focus of the paper is on detection data quality problems introduced by a shared use of data in different processes or by periodically synchronized data sources in process based systems using Web services. The methodology is explained by means of the bookshop example introduced in Section 4; the same example has been used as the basis of the development of a software application that supports the methodology. The developed tool supports both the Quality Requirements Definition phase and the Warning phase. Obviously this software is not supposed to be a complete tool to support warning management but it is a first attempt to automatically manage warning and alert messages for data quality improvement.

Future work will focus on the extension on the data error detection and recovery model and its integration in the model proposed in [7] that manages errors related to functional aspects of Web services.

Furthermore, in order to improve the automatic warning management, it will be also addressed the definition of automatic reasoning techniques that use the Warning Log database for the automatic classification of warnings and the association of the most suitable recovery actions.

ACKNOWLEDGEMENTS

This work has been partially supported by the European Project WS-DIAMOND.
REFERENCES


