

Management of Changes in Web Service Protocols

Ahmed Azough and Emmanuel Coquery and Mohand-Said Hacid

Claude Bernard Lyon 1 University, Btiment Nautibus,
8, boulevard Niels Bohr, 69622 Villeurbanne cedex France

Abstract. Web services constitute the new generation of web technologies for application integration. A web service can be considered as an application offered by a provider service and could be invoked via the web by a client service. Business Protocol notion is a very interesting formalism to represent web services in terms of interactions with other between services. Although its attractive concept of regrouping small and loosely coupled pieces of application functionality around the network improve flexibility and reach of existing IT infrastructure, the frequent and easy occurrence of change in the different components can lead to inconsistencies or errors. Business Protocol notion is a very interesting formalism to represent a web service in terms of interactions with another one. The aim of this work is to handle the change operations concerning the interaction process of a web service with represented by its business protocol and secondly to propagate this change to its interlocutors in order to guarantee the continuation of the communication.

1 Introduction

Web services are increasingly gaining importance in the development and deployment of enterprise software applications. Although their attractive concept of regrouping small coupled pieces of application functionality around the network, the frequent and easy occurrence of change in the different components can lead to inconsistencies or errors.

Nowadays economic environment is characterized by quick and easy changeability. For several reasons enterprise is frequently subject to change. Their economic success more and more depends on their ability to flexibly react on change at the market, the development, or at the manufacturing side. Enterprises that present it self as services implemented and invoked via the network can manage many relevant adaptations to improve their service. This change may concern business process as well as security and trust level. Naturally, network of web services is highly dynamic. Since facilitating change is easier, change occurs more frequently. But although the change itself is affordable, the impact of change could be unmanageable or expensive.

Generally, the partners involved in a web service conversation exchange messages following a business or conversation protocol, which can define the possible and the correct order of messages invocation between the provider and the

client. Each one of them gives its possible conversation policy via its own business protocol. If one of these interlocutors change the structure of his protocol (e.g., introduce new operation invocations or delete some of them) especially the provider service, the challenge is to decide whether this change affects the interactions with the client service and if so how to manage the impact of this change by minimizing the loss and maintaining compatibility.

After adopting a pertinent representation of web services for change detection, and resuming almost the works done on the automated management of communicating components, we have developed an algorithm for propagation of change on the web services and proofed its correctness. We adopt a comprehensive approach for the controlled evolution of web services business protocols in cooperative web services framework. First, we introduce a set of elementary operators with well-defined semantics for changing web services protocols. Second, we present a process for the automated adaptation of protocols to assure compatibility between two communicating web services.

This document is organized as following

2 Related Works

2.1 Conversational Model

Although SOA are emerging as the technology of choice for application development many challenges should be raised to a better description and development strategy of web services. Web services technology is characterized by two trends of specification; the first is based on interface description that considers both the provider and requester as web service and specify to each other how to communicate invoking or delivering an operation.

The second is the one considering web services as a set of ordered operations invoked depending to the result of each others, it allows to define the communication between a provider and a client as a complete conversation. A conversation then is defined as a business protocol with collection of business steps taking place in a prescribed manner and leading to business objective.

The conversation description model [6] and [8] is based on the state-machine formalism that fits better to the reactive behavior and conversations description facilitate web services operations specification rather than Petri nets or activity diagrams. States are labeled by conversation levels attained by the client and transitions are labeled by the names of messages corresponding to the invocation of a service operation or to its reply (e.g. Names of operations from WSDL service interface).

2.2 Change Management Approaches

The evolution control of the process choreographies in industrial workflows is a close research problem to ours. Process Choreography is defined as the ordered set of observable interactions of services with their users. In [12], Rinderle S,

Wombacher A, and Reichert M.U. addressed this challenge by first classifying changes into additive and subtractive changes from the framework dimension and into variant and invariant changes from the impact dimension. Propagating changes to the partner public process is done by what we can call the "Raw method". Applying this method to web services consists on calculating directly the differences between the new and the old version of the changing process, to add the added part for an additive change and to delete the deleted part for a subtractive change only if the change is classified as variant. Although this approach allow for an easy calculation of change in the case of web service protocol, its cost would be expensive in terms of times and technology development since it propagate all the changes occurred in the provider to the client. This one could not be interested by this changes or would not make use of the new added functionalities to its partner. Also in the case of invariant change, some server changes could be interesting to the partner and worth a their propagation. After exposing our approach, we will make a comparison between the two approaches to highlight this inconveniences.

In [9], an approach for the evolution control of access rules in cooperative systems is exposed. Authors propose a Meta model that describes organizational entities and the relations between them. They then define a prototype for change operation that occurs on an organizational model that preserves the model consistency and correctness. Thus they propose an exhaustive list for basic and high level operations on organizational models. After each operation, all former access rules should either be migrated or adapted to the new organizational model. Although the prototype of the operation of change is interesting, a joint analysis of an operations set could be more adequate to organizational model evolution that can produce a model totally different from the original one. The approach gives no method to determine elementary operations performed by comparing two versions of the organizational model.

In [14], Skogsrud H, Benatallah B and Casati F treat the problem related to the controlled evolution of access control policy, how an enterprise could modify its policy without disturbing ongoing negotiations. Each requester connecting to provider to get a service activates an instance of negotiation that will save all the levels achieved and the conditions satisfied by the client. The approach proposes to rebuild a hybrid protocol for each non compliant instance until its achievement. Although this idea could help to prevent from the deactivation of clients privileges and the reload of the clients, its efficiency is discussed. Such a decision could lead in a dynamic execution environment to hundreds or thousands of independent hybrid policies supported simultaneously by the system, which could lead to the crush of the security barrier.

3 Operation Based Approach

3.1 Description

Giving two services conversing following there business protocols, the conversation can be cut after an abrupt change occurring in one of them, and all

data exchanged between users would be lost. Our aim is to manage this impact by calculating the change produced in one protocol and projecting it to the other with regards to the other service interests. We propose the scenario that a provider/server service is conversing with a client one while a change occurs in the provider protocol, our aim is to provide the best adaptation of the client protocol in order to maintain the conversation. Some questions can be formalized to build our point of view and invent our contribution to the problem:

1. How to model a structural change in a web service protocol? We have adopted a modeling system based on elementary operations. By comparing the two version of the changed protocol, we can intuitively produce a list of elementary operations performed on the old protocol version to create the new one.
2. How to project changes to the client protocol? This is the core of our work. By using the previous list of operations and the intersection protocol between the client and server protocols we create the new client protocol.
3. Which change should be propagated to the client? In our approach, we apply a filter on changes by respecting a condition of propagation that take consideration of the client interests using its previous behaviors.

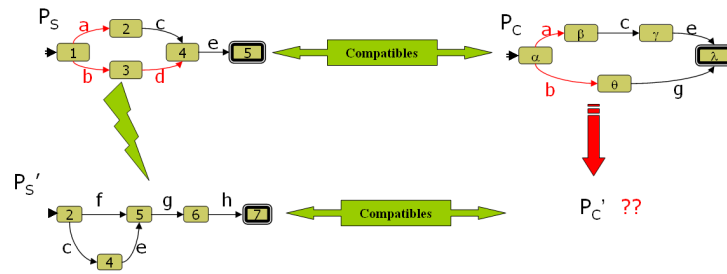


Fig. 1. Problem

3.2 Definitions

Business protocol: *Business protocol* is a formalism for web service specification that defines supported messages and their order of invocation. We have chosen to model business protocol using Deterministic Finite state machine, since deterministic schema seems to be more adequate to web service execution and facilitates their management. While

- States : represents different phases of the service conversation.
- Transitions : represent messages exchanged with other services.

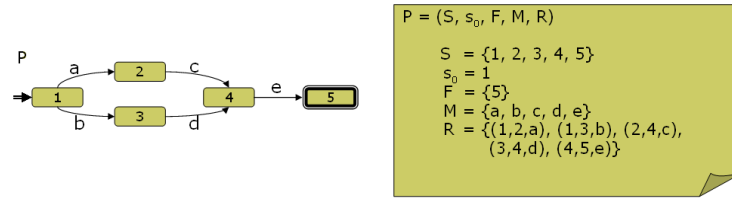


Fig. 2. BP

Execution path: An *execution path* is a transition sequence that starts from the initial state of the protocol and ends with a final state of the protocol. A *partial execution path* is an execution path except that its first and final state are not necessarily the initial and a final state of the protocol. The Execution-paths set recognized by the business protocol P_S in Figure 2 is: $EP = \{(1,2,a), (2,4,c), (4,5,e), (1,3,b), (3,4,d), (4,5,e)\}$

Conversation: A *conversation* of a business protocol is a word recognized by its associated automata. It represents a complete message exchange supported by a service following its business protocol. Due to determinism of the protocol, to each conversation c belongs a unique execution path. A *partial conversation* a conversation where the first state and the last state are not necessary the initial and a final state. The conversation set recognized by the business protocol P in Figure 2 is $CP = \{a.c.e, b.d.e\}$.

Subparts of a conversation: Let X be a conversation of the Protocol P . A subpart of X is an internal sequence of transitions of it. A subpart of P is an partial conversation. Subparts sets are defined as following:

- $Sp(X) = \{v \mid \exists v', v'' \text{ where } v'.v.v'' = X\}$
- $Sp(P) = \{v \mid \exists v', v'' \text{ where } v'.v.v'' \in CP\}$

Intersection protocol: Intersection protocol as defined in [18] is an interesting tool that defines the possible conversation that can be hold between two services. The figure shows the intersection protocol between two protocols of a server (provider) P_S and its partner a client (requestor) web service P_C .

Compatibility: We say that two protocols P_S and P_C are compatible if and only if they support some common conversations: $CP_S \cap CP_C \neq \emptyset$. Previous protocols P_S and P_C in figure 3 are compatible since: $CP_S = \{a.c.e, b.d.e\}$ and $CP_C = \{a.c.e, b.g\}$ and $CP_S \cap CP_C = \{a.c.e\}$

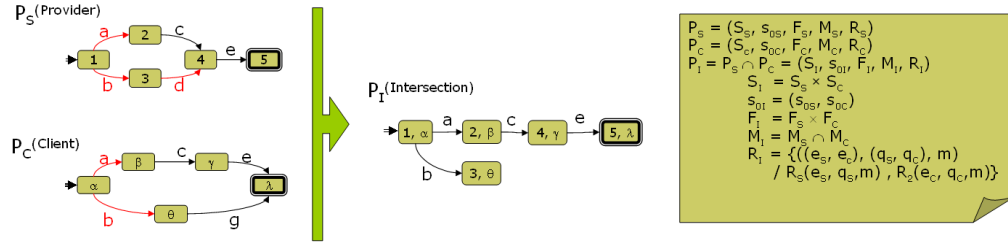


Fig. 3. Intersection Protocol

Correspondence of states and transitions: Two states $e \in P_S$ and $f \in P_C$ are correspondents if and only if $(e, f) \in P_I = P_S \cap P_C$ or if one is created by a change projection after the creation of the other. Notation: $e \Leftrightarrow f$. Two transitions $r1 = (a, b, m) \in RS$ and $r2 = (c, d, n) \in RC$ are correspondents if and only if $a \Leftrightarrow c$ and $b \Leftrightarrow d$ and $m = n$. In Figure 3 P_S and P_C have the following correspondences:

- For states: $1 \Leftrightarrow \alpha$, $2 \Leftrightarrow \beta$, $3 \Leftrightarrow \theta$, $4 \Leftrightarrow \gamma$, and $5 \Leftrightarrow \lambda$,
- For transitions: $(1, 2, a) \Leftrightarrow (\alpha, \beta, a)$, $(2, 4, c) \Leftrightarrow (\beta, \gamma, c)$, $(4, 5, e) \Leftrightarrow (\gamma, \lambda, e)$ and $(1, 3, b) \Leftrightarrow (\alpha, \theta, b)$

3.3 Change Operations

To model protocol change we define a basic operation set that allow evolution of the business protocol. Let P be a business protocol and P' its new version after a change operation. The figures 4 and describe these elementary operations.

4 Change Projection Process

Let P_S and P_C be the two Business Protocols related respectively to the server web service and the client web service. $P_S = (S_S, sO_S, F_S, M_S, R_S)$ and $P_C = (S_C, sO_C, F_C, M_C, R_C)$ After modification of the server web service, his business protocol also changes to : $P'_S = (S'_S, sO'_S, F'_S, M'_S, R'_S)$. An adaptation should be done in order to create the new client protocol $P'_C = (S'_C, sO'_C, F'_C, M'_C, R'_C)$.

The steps that the process take are the following:

1. Determine the intersection protocol between the Old Protocols.
2. Deduce the list of change operations of the server protocol by comparing its two versions.
3. Deduce the list of change operations to perform on the client protocol.
4. Calculate the new client protocol.

Operation Name	Description	Example	Formal change
Create-State (CS)	Create a new State	$P' = CS(P, s)$	$S' = S \cup \{s\}$
Delete-State (DS)	Delete a state	$P' = DS(P, s)$	$S' = S \setminus \{s\}$
Create-Transition (CT)	Create a transition	$P' = CT(P, s, s', m)$	$R' = R \cup \{(s, s', m)\}$
Delete-Transition (DT)	Delete a transition	$P' = DT(P, s, s', m)$	$R' = R \setminus \{(s, s', m)\}$
Become-Final-State (BF)	Change an state to be final or add a new state that is final	$P' = BF(P, s)$	$F' = F \cup \{s\}$ and $S' = S \cup \{s\}$
Become-Not-Final-State (BN)	Change a final state to be non final	$P' = BN(P, s)$	$F' = F \setminus \{s\}$
Change-Initial-State (CI)	Change the initial state of the automata	$P' = CI(P, s_1)$	$s_0' = s_1$

Fig. 4. Elementary Operations Description

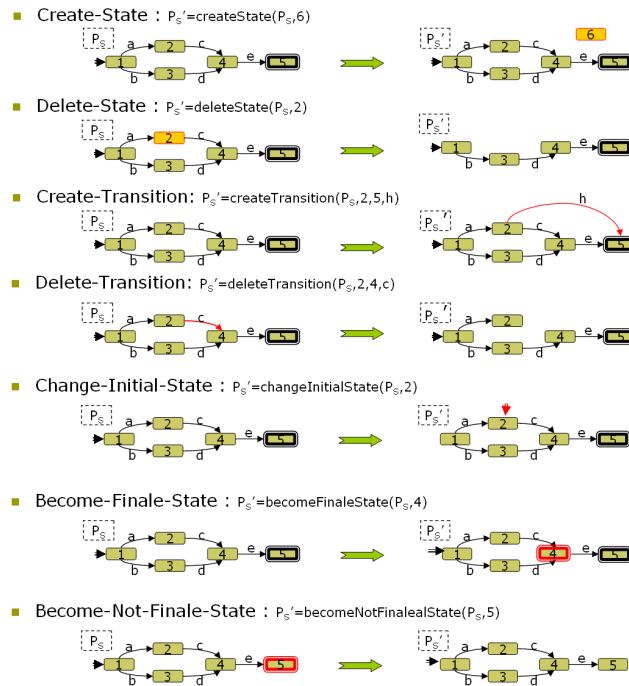


Fig. 5. Elementary Operations

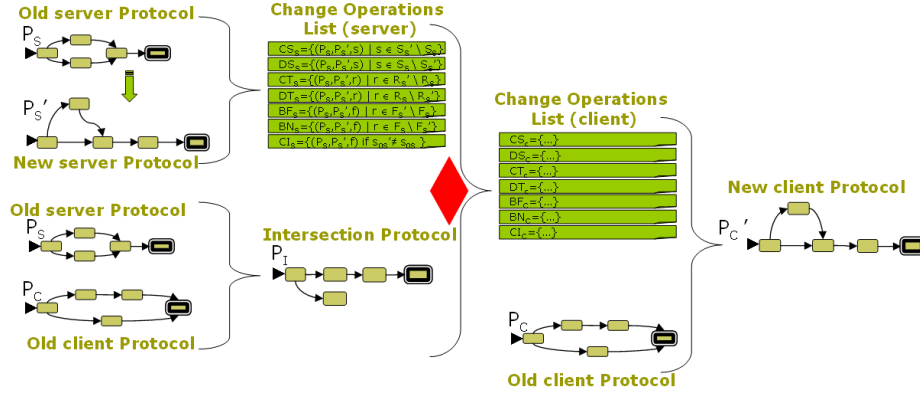


Fig. 6. Process

4.1 Determination of the Operations list

Considering the two versions of the server protocol P_S and P'_S , the determination of the batch of elementary operations performed on P_S to produce P'_S is done intuitively by calculating directly the differences between the states and transitions sets of the two protocols. The chart in figure 7 resume the formulas used for each type of operation; the global list of change basic operation is the union of all the operation lists.

4.2 Propagation Condition

To respect the interests of the client during the adaptation process of change perorations a selective propagation should be performed in order to decide which change occurring in the server is interesting for the client and which is not. In particular for new created transitions which represent new operations created in the server. In order to minimize time and effort we choose to predict beneficial new operations to the client based on its previous behavior represented in the intersection protocol.

To characterize potentially interesting new operations for the client we select in terms of conversations (cf Definitions) and we define a "*Propagateability condition*" on new conversations of the server protocol.

"Propagateable conversation": A new conversation is considered interesting to the client if:

1. all its transitions belonged earlier to the old client protocol except for the newly added transitions.
2. its final state is either a new final state in the server protocol or an old one that had a finale equivalent state in the old client protocol.

This condition is formulated as in the Figure 9

List Name	Calculation Formula
Create-State List	$CSL_S = \{CS(P_S, P_S', s) \mid s \in S_{S'} \setminus S_S\}$
Delete-State List	$DSL_S = \{DS(P_S, P_S', s) \mid s \in S_S \setminus S_{S'}\}$
Create-Transition List	$CTL_S = \{CT(P_S, P_S', r) \mid r \in R_{S'} \setminus R_S\}$
Delete-Transition	$DTL_S = \{DT(P_S, P_S', r) \mid r \in R_S \setminus R_{S'}\}$
Become-Final-State	$BFL_S = \{BF(P_S, P_S', f) \mid r \in F_{S'} \setminus F_S\}$
Become-Not-Final-State List	$BNL_S = \{BN(P_S, P_S', f) \mid r \in F_S \setminus F_{S'}\}$
Change-Initial-State List	$CIL_S = \{CI(P_S, P_S', s'_{0S}) \mid \text{if } s_{0S} \neq s'_{0S}\}$
Global Operation List $COL_S = CSL_S \cup DSL_S \cup CTL_S \cup DTL_S \cup CIL_S \cup BFL_S \cup BNL_S$	

Fig. 7. Lists

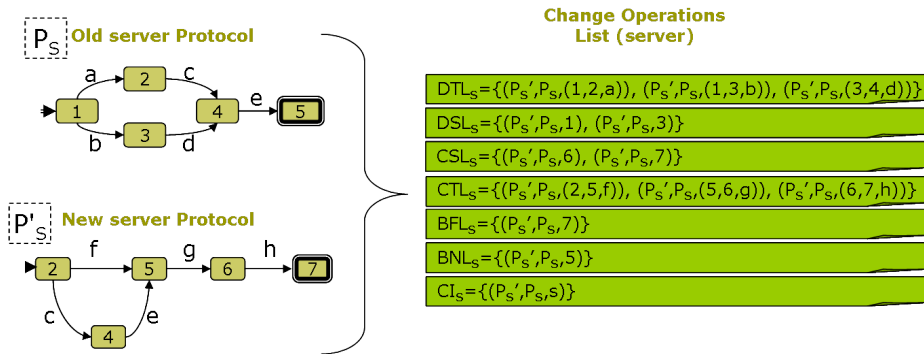


Fig. 8. Server List

- (X) is an Propagateable new conversation iff:
 - $S_p(X) \cap S_p(P_S') = S_p(X) \cap S_p(P_C)$
 - $f \in F_{S_n} \vee \exists f' \in F_C, f \Leftrightarrow f'$

Fig. 9. PropCondition

4.3 Client Change Operations List

To type of basic operation a propagation formula is used to create change operation list to perform for the client. The Figure 10 resume this formulas.

Delete Transitions Projection To propagate this change to the client protocol, we should use the delete-Transition operations list, and for each operation in the list see if the deleted transition in the server protocol has a correspondent transition in the client protocol, if so, we delete the correspondent transitions in the client protocol.

Delete State Projection To propagate this change, we should for each operation of state deletion look for the correspondent state of the deleted state in the client protocol.

Create State Projection The propagation of this operation is done by the creation of a new state in the client protocol for each new state created in the server protocol.

Create Transition Projection The projection of a 'create-Transition' operation is done following the edges of the created transition.

- If the two states belong to the old server protocol and have correspondents in the old client protocol, we create new transitions in the client protocol between the correspondent states.
- If one of the states that limit the transition is new and the other is old, we create transition between the new state created in the client protocol and the correspondent state of the old states in the client protocol.
- If the states that limit the transition are new in the server protocol, we create a new transition between the new states in the client protocol.

Change Initial state projection The projection of the change initial state operation is done by looking for the correspondent state to the server new initial state in the client protocol and naming it as the new client initial state. Many cases can be treated:

1. the server new initial state is a new state in the protocol, in that case the new correspondent state in the client protocol would be the client initial state.
2. the second case is when the server new initial state is an old state in the protocol
 - (a) If the state has a unique correspondent state in the client protocol this one state would be the new initial state for the client protocol. This is correct even if the new state has been newly created.

- (b) If the new server initial state has many correspondent states in the client protocol, the solution is not direct. We can opt for other three possible choices: like determinization of protocol a transition count method or a Statistic method

Become Final State projection The projection of a 'Become Final State' is done by performing become final operation on all the correspondent state to the becoming final state in the client protocol.

Become Not Final State projection The projection of a 'Become Not Final State' is done by performing become Not final operation on all the correspondent state to the becoming not final state in the client protocol.

Operation	Propagation Formula
Delete-Transition	$DTL_c = \{(P_c', P_c, (e_c, e_c', m)) \mid ((e_s, e_c), (e_s, e_c'), m) \in S_I \text{ and } (P_s', P_s, (e_s, e_s', m)) \in DTL_s\}$
Delete-State	$DSL_c = \{(P_c', P_c, e_c) \mid (e_s, e_c) \in S_I \text{ and } (P_s', P_s, e_s) \in DSL_s \text{ and } \{t \in S_s \mid (t, e_c) \in S_I\} \setminus \{e_s\} = \emptyset\}$
Create-State	$CSL_c = \{(P_c', P_c, s) \mid (P_s', P_s, s) \in CSL_s\}$
Create-Transition	$CTL_c = \{(P_c', P_c, (a', b', m)) \mid (P_s', P_s, (a, b, m)) \in CTL_s \text{ and } ((\exists (a, b) \in S_s \mid ((a, a'), (b, b')) \in S_I) \text{ or } (\exists a \in S_s \mid (a, a') \in S_I \text{ and } (P_s', P_s, b') \in CSL_c) \text{ or } (\exists b \in S_s \mid (b, b') \in S_I \text{ and } (P_s', P_s, a') \in CSL_c) \text{ or } ((P_s', P_s, b') \in CSL_c \text{ and } (P_s', P_s, a') \in CSL_s))\}$
Become-Final-State	$BFL_c = \{(P_c', P_c, s') \mid ((P_s', P_s, s) \in BFL_s \text{ and } (s, s') \in S_I) \text{ or } ((P_s', P_s, s') \in CSL_s \text{ and } s' \in F_s)\}$
Become-Not-Final-State	$BNL_c = \{(P_c', P_c, s') \mid (P_s', P_s, s) \in BNL_s \text{ and } (s, s') \in S_I \text{ and } \{t \in F_s \mid (t, s') \in F_I\} \setminus \{s\} = \emptyset\}$
Change-Initial-State	If $((P_s', P_s, s) \in CSL_s)$ $CIL_c = \{(P_c', P_c, s)\}$ Else If $(\exists s' \in S_c \mid (s, s') \in S_I)$ so $CIL_c = \{(P_c', P_c, s')\}$ Else "Non Deterministic protocol, use another method"

Fig. 10. PropageFormula

4.4 Example

5 Approach limits

Despite its importance our approach has some limits. This limits comes from our choice to represent the business protocol as a deterministic finite state machine. in fact and during propagation of the change operations the client protocol could become non deterministic.

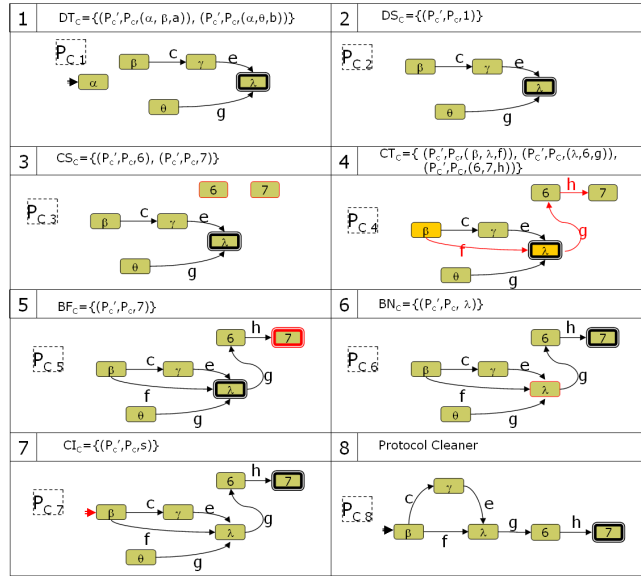


Fig. 11. Process Steps

5.1 Create transition propagation Limit

Let create-Transition $P'=(P,(2,5,f))$ If the client protocol contains a transition f that goes from a state that corresponds to the state 2 and leads to a state non corresponding to 5, propagating this transition makes the client protocol non determinist as depicted in Figure 12. A test before propagation of such a transition or a determinisation of the protocol after the propagation can solve the problem.

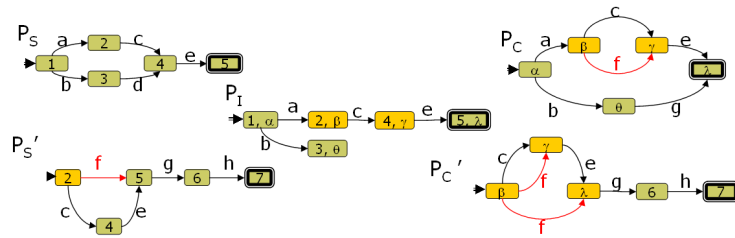


Fig. 12. limit1

5.2 Create transition propagation Limit

If the new initial state in the server protocol has many corresponding states in the old client protocol, propagation this change makes the client protocol non-deterministic. After the propagation of such a change a determinisation of the protocol is necessary, we can also use statistical method to choose the new client initial state between old ones.

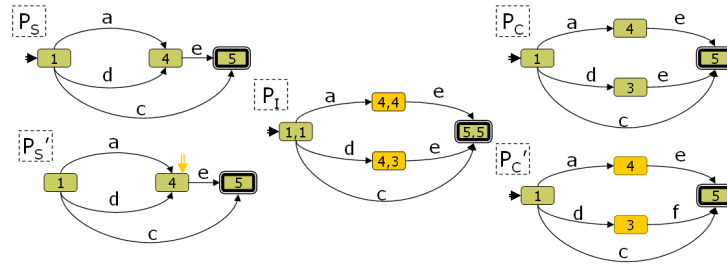


Fig. 13. limit2init

6 Conclusion and Perspectives

The management of the impact of change in web service is a critical paradigm that should be well studied and resolved in order to strengthen the service architecture frameworks and minimize the fall of service oriented systems. Many approaches were developed in order to control the evolution of web services but yet their efficiency and power is discussed. During my internship, we focused my supervisor and me on the adaptation of conversations when occurs a structural change in a web service protocol. We developed a comprehensive approach for the controlled evolution of web services business protocols in cooperative web services framework. Our work has two important contributions:

1. Proposing a model with elementary and well defined semantics operation to formalize the change of web service protocols.
2. Developing a new approach for adapting a client to the evolution of a server by to maintain compatibility and converse-ability between them with regards to client interests.

Important contribution of our work was to adopt a selective propagation of new conversations added to the server. Such an approach helps as to predict the client decision in implementing new operations. Another challenging issue that we are working on and that will be developed in another separate contribution is the protocol adaptation in ad hoc manner. How should we migrate active

conversations during their execution the change occurs? And how could we make use of the data and parameters already exchanged between the client and the server before the changes? Should we abort all conversations and reload them to begin from the initial state of the new protocol? Another future work that could be interesting to do is the adaptation following a new extended definition of compatibility. In fact we could enlarge our definition of compatibility by imposing that so that two systems be compatible, any message sent by a partner should have a response by the other, by this way, we will never have an error message for non conformity or non support. We should for this extend also the definition of our business protocol to distinguish from incoming and outgoing transitions and to enlarge our propagation method to take this in consideration.