Doing and undoing in the framework of Web services

Marie-Claude Gaudel
LRI, Univ. de Paris-Sud & CNRS
(Programming and SE group)

Thanks to:
Michel Beaudouin-Lafon and Stéphane Conversy, LRI (Man-Machine Interaction group)
and to Valérie Issarny (INRIA), Nicole Levy (Univ. de Versailles), and the other IST DSoS project members

This talk is not on testing ☺

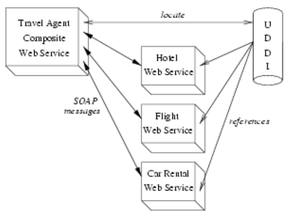
- A. Denise, M.-C. Gaudel, and S.D. Gouraud. **A generic method for statistical testing.** In *Proceedings of the 15th IEEE International Symposium on Software Reliability Engineering (ISSRE 2004)*, pages 25-34, Saint-Malo, France, November 2004.
- S.-D. Gouraud, A. Denise, M.-C. Gaudel, and B. Marre. **A new way of automating statistical testing methods**. In *Proceedings of the 16th IEEE International Conference on Automated Software Engineering (ASE 2001), 26-29 November 2001, Coronado Island, San Diego, CA, USA*, pages 5-12. IEEE Computer Society, 2001.
- M.-C. Gaudel. **Problems and methods for testing infinite state machines**. In *VI Workshop de Métodos Formais (WMF 2003)*, volume 95, Campina Grande, BR, October 2003. Electronic Notes in Theoretical Computer Science (ENTCS).
- G. Lestiennes and M.-C. Gaudel. **Testing Processes from Formal Specification with Inputs,Outputs and Data Types**. In *Proceedings of the 13th International Symposium on Software Reliability Engineering (ISSRE 2002), 12-15 November 2002, Annapolis, MD, USA*, pages 3-14. IEEE Computer Society, 2002.
- Gregory Lestiennes, Marie-Claude Gaudel . **Test de systèmes réactifs non réceptifs,** MSR, Autrans, Oct. 2005 (english version available)

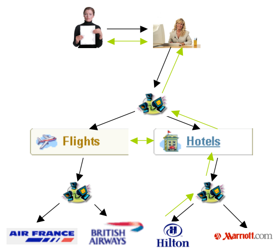
http://www.lri.fr/asspro/membres/gaudel.fr.html

Dependable composition of web services

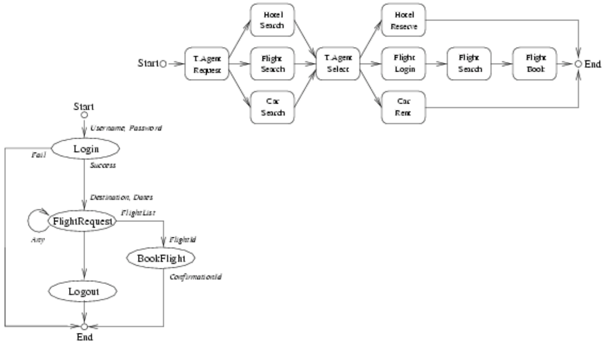
- *Component web services are shared with an unknown population of other users*
- *They are managed by independent entities*
- *Composite web services cannot :*
  - lock a component web service for a long time
  - rely on roll back or backward recovery when something goes wrong (impossibility to successfully complete a composed operation, or crash of one component, …)
  - assume that some «pre-commit» feature is available in a component web service

The travel agent case study





Example of «transactions»



Aim and outline of the talk

- *Composition of Web services raises some problems of* ***atomicity***
  - There is a need to undo some sub-operations when a "composed transaction" cannot succeed
- *Similarities with "Undo" in collaborative work (for instance group editors)*
- ***Presentation outline***
  - Good old "undo" and "redo"
  - "undo" and "redo" in group editors
  - Transposition to composite Web services

What is undo? ☺

un·do v

1. vti to open, unfasten, untie, or unwrap something

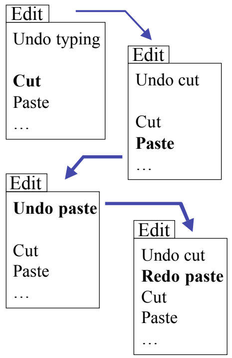
2. vt to cancel or reverse the effect of an action

3. vt to cancel the effect of the last command or action done on a computer, restoring the material being worked on to its previous condition

4. vt to bring somebody or something to ruin or disaster ☹

Encarta® World English Dictionary © 1999 Microsoft Corporation. All rights reserved. Developed for Microsoft by Bloomsbury Publishing Plc.

Undo-Redo: the good old style



- *"linear undo":*
  - Only the previous action is undoable
  - And then the previous one, etc
  - => stacks of possible "undo.s" and "redo.s"
- *some actions are not undoable in some states*

Bases of the implementation

- *History buffer + "redo" stack*

  O1 O2 O3 O4   O4 O3

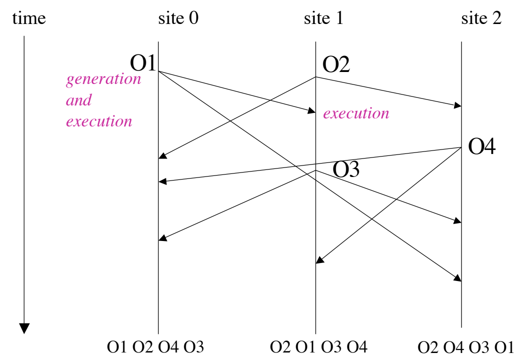
- Execution of a new command => *addition to the history list*
- Undoing the last action of the history list => *moving it to the "redo" stack*
- Redoing the top action of the Redo stack => *moving it to the history list*
- *"Undo" and "Redo" do not appear in the history list (meta-actions)*
- *Undo can be performed via **state recovery** or **reverse operations***

animation!

Non Linear Undo

- *A nice wish : any past action is undoable … "if that is meaningful"* ☺☺☺
- *Interlisp (1975)*
  - "The user is explicitly warned that nonlinear undo might have unpredictable effect"
- *"Selective undo" (Berlage 1994): the user is not able to select "undo" of a command when "it does not make sense" (?).*
  - Collaborative graphic editor: GINA system
- *"Undo any operation at any time in group editors" Chengzeng Sun, Proc. of 2000 ACM conf. On Computer-Supported Cooperative Work*
  - **REDUCE system**

Collaborative editing: a scenario

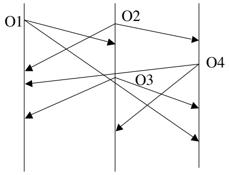


Collaborative editing: consistency

- ***Causal ordering** relation (dependent operations)*
  - O_a generated at site i, O_b generated at site j, $O_a \rightarrow O_b$ iff
    - i =j and O_a generated before O_b
    - i≠j and the execution of O_a at site j happened before the generation of O_b
- *Independent operations*
  - neither $O_a \rightarrow O_b$ nor $O_b \rightarrow O_a$: $O_a \parallel O_b$
- ***Intention** of an operation*
  - The intention of an operation O is the execution effect that can be achieved when applying O to the state from which O was generated
- *Consistency*
  - *Convergence* (same state after the same set of operations), and *causality preservation* (time stamping), and *intention preservation*

Back to the example

- *O_1 □ O_3 and O_2 □ O_3 and O_2 □ O_4*
- *$O_1 \parallel O_2$, $O_1 \parallel O_4$ and $O_3 \parallel O_4$*
- *More about independent operations*
  - assume as initial state «abc»
  - O_1 is *Insert[2,X]* => intention: «aXbc»
  - O_2 is *Insert[3,Y]* => intention: «abYc»
  - **Global intention:** «aXbYc»
  - Site 0: O_1 O_2… «aXYbc» => intention violation ☹



- *Solution for **intention preservation**: Operational transformations*
  - Site 0: O_1 **O'_2**…, with O'_2 = Insert[4, Y]
  - O'_2 is the result of the so-called Inclusion Transformation IT(O_2, O_1)

Transformations (Sun & al, 98, 2000)

- *Inclusion Transformation of O_a against O_b:*
  - IT(O_a,O_b) transforms O_a into O'_a, in such a way that the impact of O_b is included in the new parameters of O'_a
- *Exclusion Transformation:*
  - ET(O_a,O_b) transforms O_a in such a way that the impact of O_b is excluded from the new parameters of O'_a
  - **Example**: $O_1 \parallel O_4$ but IT(O_4,O_1) is not sufficient at site 0
  - O_1 and O_4 are generated at different states, because of the execution of O_2 at site 2 before O_4
  - At site 0, when arriving after O_1O_2,
    - O_4 is transformed into O'_4 = ET(O_4,O_2), because O_2 □ O_4 ,
    - and then into IT(O'_4, O_1), because $O_1 \parallel O_4$

Some Technicalities

- *To make a long story short…* IT and ET must be defined for any couple of basic operations. Very often the result is the identity. IT and ET are application dependent.
- *Context of an operation O:*
  - CT_O , list of operations needed to bring the system from some initial state to the state on which O is defined
- *"context equivalent" relation*
  - O_a ✧ O_b □ $CT_{Oa} = CT_{Ob}$
- *"context preceding" relation*
  - O_a ↪ O_b □ $CT_{Ob} = CT_{Oa}+[O_b]$
- *Reversibility requirement*
  - if O_a ✧ O_b, then O_a = ET(IT(O_a, O_b), O_b)
  - if O_a ↪ O_b, then O_a = IT(ET(O_a, O_b), O_b)

This was «doing», what about «undoing» ?

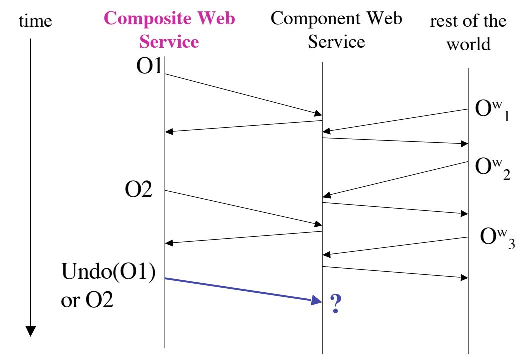
- *Let* Undo(O_i) *generated or received at site k, with history buffer HB_k* = $O_1…O_iO_{i+1}…O_n$
- *Assumption: for any O, there is a reverse operation $\underline{O}$*
  - Reminder : backward recovery cannot be assumed
- ☺ *i = n : execution of $\underline{O}_n$*
- ☹ *i < n : execution of $\underline{O'}_i$ obtained by transformation of $\underline{O}_i$ such that:*
  - ***$O_1…O_iO_{i+1}…O_n\underline{O'}_i$* has the same effect as *$O_1…O_i\underline{O}_iO'_{i+1}…O'_n$***
  - Roughly, the transformation of $\underline{O}_i$ into $\underline{O'}_i$ includes the impacts of *$O_{i+1}…O_n$* , and the transformation of *$O_{i+1}…O_n$* into *$O'_{i+1}…O'_n$* excludes the impact of *O_i*
- ***execution of $\underline{O'}_i$ and then update of HB_k***

animation !

Transformations for undoing

- ***Just the same ones as those for doing!***
- *Note that the strategy above is equivalent to doing $\underline{O_i}$, with*
  - O_x □ $\underline{O_i}$ for $1 \leq x \leq i$, and
  - $\underline{O_i} \parallel O_x$ for $i+1 \leq x \leq n$
- $\underline{O'_i}$ ***= LIT(***$\underline{O_i}$ ***, HB***$_k$***[i+1, n])***
  - where LIT is the generalisation of IT to lists of operations
- ***NB:*** *the new* HB_k *is not* $O_1 \ldots O_i O_{i+1} \ldots O_n \underline{O'_i}$ *but* $O_1 \ldots \mathbf{O_i * } O'_{i+1} \ldots O'_n$
  - This allows an elegant and efficient treatment of *Redo*(O_i)
  - See Sun & al. papers… not needed for web services

Transposition to Web services



Slightly simpler than in collaborative editing

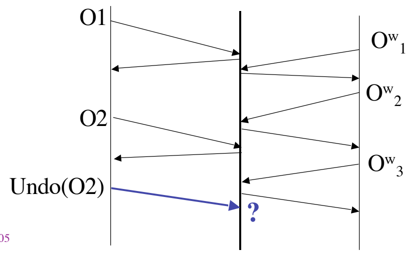
- *No problem of causal ordering: "doing" is straightforward (in first approximation…)*
- *"undoing" could follow the transformational model*

Undo any operation at any time in Web Service?

- *Requirements (**transactional attitude** of composable Web services, similar to Mikalsen & C°)*
  - All undoable "operations" in the CWS are reversible in their WS
  - There is a unique history buffer for each WS, at least when "composite transactions" are performed
  - IT and ET are defined
- *Back to the example: HB =* $O_1 \; O^w_1 \; O^w_2 \; O_2 \; O^w_3$*, and then* Undo(O2)…
  - Execution of **IT($\underline{O_2}$, O^w_3)**
  - Modification of HB into $O_1 \; O^w_1 \; O^w_2 \; O_2^* \; ET(O^w_3, O_2)$
  - *see next slide*

Back to the example

- *HB* = $O_1\ O^w_1\ O^w_2\ O_2\ O^w_3$
- Undo(O2)
  - Execution of **$IT(\underline{O_2}, O^w_3)$**
  - Modification of HB into $O_1\ O^w_1\ O^w_2\ \mathbf{O_2^*\ ET(O^w_3, O_2)}$



Some research issues

- *It gives a nice general model. How to instantiate it?*
- *The "Travel Agent" case study*
  - O_2 is some flight reservation
  - O^w_3 is another reservation for the same flight
    - which has been satisfied => $IT(\underline{O_2}, O^w_3) = \underline{O_2}$
    - which is in a waiting list => $IT(\underline{O_2}, O^w_3)$ satisfies the 2nd reservation
      - O'^w_3 is then a successful reservation
- *Wanted: a definition of "has the same effect as" in*
  - $\mathbf{O_1 \ldots O_i O_{i+1} \ldots O_n \underline{O'_i}}$ **has the same effect as** $\mathbf{O_1 \ldots O_i \underline{O_i} O'_{i+1} \ldots O'_n}$
  - Observational equivalence of states…
  - to be extremely flexible… Actually, it may not be an equivalence
    - The waiting list was full: O^w_3 will not be satisfied, even if O_2 is undone…
  - Strong relation with intention preservation

Some other issues

- *It works when transactions are not too long, the possible operations are not too numerous, the transformations do not take too much time (lock of the site during the transformations…)*
- *Possibility of providing generic "wrappers" for making web services composable ?*
- *If interested:*
  - *« Toward undoing in Composite Web Services », in Architecting Dependable Systems III, pp. 59-68, LNCS 3549, 2005*

Formalisation

- A. Imine, P. Molli, G. Oster, and M. Rusinowitch. « Proving correctness of transformation functions in real-time groupware ».
- In K. Kuutti and al., editors, *Proceedings of the 8th European Conference on Computer Supported Cooperative Work, 14-18 September 2003, Helsinki, Finland*, pages 277-293. Kluwer, 2003.