CS 578 – Software Architectures
Fall 2013
Homework Assignment #3
Due: *Thursday, November 14, 2013*
– see course websites for submission details –

Background

As you have experienced for the last two weeks, while software modeling provides powerful ways to assess design decision rationales, it could also be costly. Configuring the domain-specific modeling environment requires considerable amount of effort. Furthermore, constructing a model of a large and complex modern software system could be highly time-consuming, even for a large team of software architects.

Labor of software architects is the most influential factor of software modeling cost as software modeling is a highly human-intensive activity. There do exist many other factors; however, those are nominal or one-time. For example, the institutional administrative cost and the utility cost to maintain workplace could be considered relatively not as significant, and the prices for buying necessary hardware and the modeling software would not be recurring as much. The labor cost of the software architects, on the other hand, could be dominant especially because the architects are often highly experienced software engineers who are at high positions that make impactful decisions when designing software systems.

Software modeling cost is not little, and it could become even more expensive when it involves more than one software architect collaboratively making design decisions. Collaborative work is a very common problem that exists not just in software modeling but in many other areas. Researchers have made attempts to solve the problem with the aid of computers (the area is called CSCW -- Computer Supported Cooperative Work), and collaborative software development (that includes modeling) is an area that receives much attention from the research community.

Upon configuring a collaborative software modeling environment, certain decisions need to be made such as (1) via what media design decisions and architectural knowledge are exchanged and (2) how to detect and resolve any potential inconsistencies. Although some might prefer to use traditional communication media such as emails or shared storages (e.g. cloud storages), the use of a proper version control system (VCS) could reduce the burden of managing model integration and the consequent conflict handling.

There are many existing VCSs that could be used for collaborative software modeling, and those could be categorized using the five criteria below:

1. The format of the data the VCS is designed to manage (text vs. graph),
2. The merging paradigm (state-based vs. operation- or change-based),
3. The rate of pushing changes (periodic vs. realtime),
4. The rate of pulling changes (periodic vs. realtime), and
5. The rate of model analysis (on-demand vs. realtime).

A large portion of the existing VCSs, however, may not be ideal in terms of the way the design decisions are exchanged. Firstly, it is often the case for VCSs that target collaborative software programming such as Subversion and Git, nevertheless of the fact that it is not impossible to use them for

collaborative software modeling, may not be suitable because those tools implement line-based text merging which does not incorporate the syntax and semantics of software models, and as a result, could invalidate the models [1]. Secondly, state-based merging is often adopted in practice for its general applicability while operation- or change-based merging has clear advantage over state-based merging [2].

Most, except a few, existing VCSs push and pull changes periodically upon requests from users, and as a consequence, they expose the users to risks of having conflicts [1]. For example, if such a VCS were being used, the changes made by a software architect would be sent to her/his colleagues only when the software architect pushes (or commits) and the colleagues pull the changes (or updates their local data) afterwards. In other words, software architects using such VCSs become not "aware" of the changes that the other software architects have made. Conflicts could occur if the software architects keep making design decisions without knowing each other's changes.

Among the numerous attempts to mitigate the side-effects of collaborative work [3], one of the most recent and promising approach is to provide workspace awareness. Workspace awareness is "the up-to-the-minute knowledge of other participants' interactions with the shared workspace" [4]. By knowing what others have been doing, conflicts may be avoided in the first place or become cheaper to be resolved [5]. There have been several trials of workspace awareness for collaborative software programming [6-10], yet the reported progress for collaborative software modeling has not been as prominent.

CoDesign is a highly extensible collaborative software modeling framework (http://softarch.usc.edu/~ronia/codesign/) that is developed at USC. Its key feature is the extensibility that allows users to be able to easily "plug in" new features and configure the collaborative modeling environment themselves. CoDesign XTEAM is an instance of CoDesign that integrates XTEAM and provides workspace awareness. Recalling the five criteria of collaborative software modeling environment, CoDesign XTEAM implements a novel combination of the features that has not existed to date.

- CoDesign is an operation-based VCS, and it works well with graphical model data.
- CoDesign pushes new changes as they are made in realtime.
- CoDesign pulls new changes only when asked.
- CoDesign analyzes the model in realtime.

CoDesign XTEAM not only runs analyses on the models that each architect maintains on her/his machine, but it also maintains "future" versions that would be created when integration happens and runs analyses on them so the architects would be able to know of the potential conflicts they may have, as projected in this paper [1].

Overview

In this assignment, you will experience more realistic software modeling and be exposed to a cutting-edge approach of collaborative software modeling. You will first perform a set of individual modeling tasks. You will then, as a team of two, perform two sets of modeling tasks collaboratively, with and without workspace awareness provided to you.

You will use XTEAM as you did for Assignment #2 for the individual task, and you will use CoDesign XTEAM for the collaborative tasks. More information regarding how to install, configure, and use CoDesign will be in the CoDesign manual document that will be provided separately later along with the description of the collaborative tasks.

Team Formation

You will be working as a team of two for the collaborative modeling tasks. Find one student within the class, and email to both Reza (gsafi@usc.edu) and Jae (jaeyounb@usc.edu) with the (1) name, (2) USC ID, and (3) email address of you and your partner by 11:59 PM November 1st.

If you do not have a partner, email Reza and Jae that you do not have one by 11:59 PM November 1st (however, the earlier the better). We will assign a random partner for you. If you are a DEN student, letting us know in which timezone you are would help us to find you a partner in the same or nearest timezone with you.

Once the team formation is complete, you will be assigned with a team number. You will perform different tasks according to your team number, whether it is an odd number or an even number.

Tasks

The individual modeling tasks are more free-form than the tasks of the previous assignment. Perform modeling, and submit (1) the resulting model along with (2) the description of your model (e.g. what each component and connector does, what events are exchanged, and how components behave upon reception of the events, etc.) and (3) the answers to the questions in the task description.

There are four sets of collaborative modeling tasks, and you will be performing two of them depending on your team number. Each collaborative modeling task is designed to take approximately half an hour (total of one hour). You do not need to practice the tasks a priori since the time spent is not a part of the grading criteria. After you complete the tasks, discuss how workspace awareness affected your collaborative modeling experience in your report.

Collaborative Modeling Sessions

You and your partner will perform the collaborative modeling tasks simultaneously using CoDesign XTEAM (CoDesign from hereon). There will be two sessions (each approximately for half an hour) for the two sets of tasks assigned to your team. You do not have to be collocated as CoDesign runs on the Internet.

You have to find a list of 5 time slots that both of you can simultaneously work from the table below. Discuss with your partner, and pick ones that have either "J" or "R". If the slot does not have any of those letters, that slot is not available. Prioritize the 5 choices, and email us the list. We will do our best to satisfy your preferences.

Reza and Jae will administer the sessions by handling the CoWare Server and the XTEAM Engines for your team. You will be provided with the IP address and the port number to which you are supposed to connect using CoWare Client at the beginning of your session. Do NOT try connecting if it is not your scheduled session. Modify the config.properties file accordingly before the session begins.

Collaborative Modeling Sessions Timetable

All times are in Pacific Time (LA Time)

From	To	11-4	11-5	11-6	11-7	11-8	11-9	11-10	11-11
9 AM	11 AM	J	J	J	J	J	J	J	J
11 AM	1 PM	J,R	J,R		J,R	J	J	J,R	J,R
1 PM	3 PM								
3 PM	5 PM	J,R		J,R		J	J	J,R	J,R

5 PM	7 PM	J,R	J,R	J		J	J	J,R	J,R
7 PM	9 PM	J	J	J		J	J	J,R	J,R

Summary

By the end of	Action Items
November 1	Email us with (1) names, (2) USC IDs, and (3) email addresses of your team, or just yours if you don't have a partner
November 3	Email us 5 available times (with priority) for your collaborative session
November 11	(1) The resulted model from the individual task (2) A report for this assignment

Tasks in Detail

This assignment consists of two parts: the individual modeling and the collaborative modeling.

Part 1: The Individual Modeling

As described above, you have more freedom than you did for the previous assignment. Make your own decisions for certain things that are not defined in this task description (e.g. attribute values, etc.), and explain them in your report.

In the given NGCA model, the data transformation is done at AmazonCloud, and it takes 2 units of time for execution. Also, the Group with which AmazonCloud is associated has 100 threads. Suppose that the transformation takes much more time (6 units of time), and the number of threads went down (3 max. threads). What happens to SalinityApp components and TempApp components?

Model AmazonCloud in more detail. Add a new component called SlaveMachine that receives transformation requests from AmazonCloud, performs the transformation, which also takes 6 units of time, and sends back the result. Make a few replica of the new component, and connect them to AmazonCloud. Create one Group and one Host per SlaveMachine, and give them maximum of 4 threads. You may choose any reasonable routing policy (e.g. Round-Robin or uniform random) to select to which SlaveMachine AmazonCloud sends the request. What happens to the simulation result as you add more SlaveMachines? How many SlaveMachines would you need to achieve the same level of service (especially the latency at SalinityApp and TempApp) as it was when the transformation only took 2 units of time?

Tip: if you want to generate a uniform random number in the simulation, use this code:

Declaration: *double NewRandom::uniform(double from, double to)*

Example: *NewRandom::uniform(0, 5)*

The uniform() function returns a uniform random number that is equal to or greater than the *from* argument and less than the *to* argument.

Part 2: The Collaborative Modeling

Task Sets of Part 2

You will receive the tasks in a few days in a separate email from the TA.

Questions

Should you have further questions, send an email to both

- Reza Safi (gsafi@usc.edu) and
- Jae young Bang (jaeyounb@usc.edu)

References

[1] Jae young Bang, Daniel Popescu, and Nenad Medvidovic. “Enabling Workspace Awareness for Collaborative Modeling.” Presented at the Future of Collaborative Software Development at Computer Supported Cooperative Work (FutureCSD 2012), Seattle, WA, February 2012.

[2] Kerstin Altmanninger, Martina Seidl, and Manuel Wimmer. “A Survey on Model Versioning Approaches.” International Journal of Web Information Systems, 5, 271–304. doi:10.1108/17440080910983556

[3] Anita Sarma and David Redmiles, “Categorizing the Spectrum of Coordination Technology.” IEEE Computer, June 2010

[4] Carl Gutwin and Saul Greenberg, “Workspace Awareness for Groupware.” Conference Companion on Human Factors in Computing Systems (CHI 1996) Vancouver, BC, April 1996

[5] Anita Sarma, David Redmiles, and Andre van der Hoek. “Empirical Evidence of the Benefits of Workspace Awareness in Software Configuration Management.” In proceedings of FSE 2008, 113–123.

[6] Jacob T. Biehl, Mary Czerwinski, Greg Smith and George G. Robertson, “FASTDash: a visual dashboard for fostering awareness in software teams.” In proceedings of CHI 2007.

[7] Yuriy Brun, Reid Holmes, Michael Ernst, and David Notkin. “Crystal: Precise and Unobtrusive Conflict Warnings.” In proceedings of ESEC/FSE 2011.

[8] Prasun Dewan and Rajesh Hegde. “Semi-Synchronous Conflict Detection and Resolution in Asynchronous Software Development.” In proceedings of ECSCW 2007.

[9] Anita Sarma, Zahra Noroozi, and André van der Hoek, “Palant´ır: Early Detection of Development Conflicts Arising from Parallel Code Changes.” IEEE Transactions on Software Engineering 2011.

[10] Jan Wloka, Barbara Ryder, Frank Tip, and Xiaoxia Ren. “Safe-commit Analysis to Facilitate Team Software Development.” In proceedings of ICSE 2009.