CS 578 – Software Architectures
Spring 2014
Homework Assignment #3

Due: *Monday, March 31, 2014*

– see course websites for submission details –

Background

The increasingly complex modern software systems are often developed by engineering teams that consist of a number of members. In such an environment, stakeholders who design the system face costly coordination challenges such as making design decisions that conflict with each other. There have been previous attempts to mitigate the problem, but uncertainty still persists in practice.

Software architects coordinate to make design decisions, and produce artifacts such as software models as a result. While traditional communication media such as phone call, email, or shared storages (e.g. cloud storages) are still being used, mature software companies have widely adopted version control systems (VCSs) and used them in concert for the purpose of reducing the burden of managing model integration and the consequent conflict handling.

However, there are several reasons why a large portion of the existing VCSs may not be ideal to be used for software models. First, it is often the case that the VCSs primarily aim at supporting collaborative software programming such as Subversion (SVN) and Git implement line-based text merging which does not incorporate the syntax and semantics of software models. As a result, they could invalidate the models during merging [1]. Second, state-based merging that accumulates stored states (e.g. files) is often adopted in practice for its general applicability while operation- or change-based merging has clear advantage over state-based merging for constructing software models [2].

Another issue with today's software design environments is that they expose software architects to the risk of making design decisions that conflict with each other. The current generation of software model VCSs synchronizes changes and identifies conflicts only in an on-demand fashion. Consequently, software architects have to make changes to the model without fully understanding what may happen when they integrate each other's changes. The cost of resolving conflicts often gets aggravated when the conflicts are found late. It is also possible that new changes made after a conflict has been introduced need to be reverted in the process of resolving the conflict, which would result in wasted effort [3].

Among the numerous attempts to mitigate the side effects of collaborative work [4], one of the most recent and promising approaches is to provide workspace awareness. Workspace awareness is "the up-to-the-minute knowledge of other participants' interactions with the shared workspace" [5]. By knowing what others are doing, conflicts may be avoided in the first place or become cheaper to be resolved [6]. There have been several trials of providing workspace awareness for collaborative software implementation with promising results [7-10].

More recently, a new approach called speculative analysis has been introduced, which "anticipates actions a developer may wish to perform and executes them in the background" [11]. Speculative analysis, when applied for collaborative software implementation, can proactively detect code-level conflicts and provide the information to developers. Speculative analysis can relatively be less obtrusive to the users than other similar techniques that proactively detect conflicts because it executes actual conflict detection rather than estimation of potential conflicts, and thus, does not report false positives.

While the existing approaches such as workspace awareness and speculative analysis have shown with how conflicts can be dealt in software development in general, yet the reported progress for their applicability in collaborative software design has not been as prominent.

The Framework for Logging and Analyzing Modeling Events (FLAME) is an operation-based collaborative software modeling framework that is being developed at USC. Its key feature is to immediately detect design conflicts in a proactive fashion and unobtrusively delivers the conflict information to software architects. FLAME not only analyzes the models that each architect maintains on her/his local machine, but it also composes and analyzes the "future" versions that may be created when integration happens so the architects would be able to be aware of the potential conflicts, as projected in this article [1].

Assignment Overview

In this assignment, you will experience more realistic software design and modeling and be exposed to a cutting-edge approach in collaborative software modeling. You will first perform an individual modeling task with XTEAM. You will then, as a team of two, perform two sets of modeling tasks collaboratively with FLAME, with and without proactive conflict detection provided to you.

Part 1: The Individual Design and Modeling

Task Details

The individual modeling task is more free-form than the tasks of 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 maximum threads). What happens to SalinityApp components and TempApp components?

You will be modeling AmazonCloud at more detailed level. 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.

Deliverables

Perform design and modeling, and submit the followings:

- The resulting model file (the .mga file)
- Description of your changes (e.g. what each newly created component and connector does, what events are exchanged, and how components behave upon reception of the events, etc.)
- Answers to the questions in the task description above.

Part 2: The Collaborative Design and Modeling

Team Formation

You will be working as a team of two for the collaborative modeling tasks. Find one partner from the class, and email to both Margi (margipat@usc.edu) and Jae (jaeyounb@usc.edu) with the (1) name, (2) USC ID, and (3) email address of you and your partner by noon March 7th.

If you do not have a partner, email Margi and Jae that you do not have one by noon March 7th (however, the earlier the better). We will assign a random partner for you. If you are a DEN student, letting us know in which time zone you are would help us to find you a partner in the same or nearest time zone with you.

Selecting Session Time

You and your partner 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 the ones that have "J" in the table. If the slot does not have the letter, that slot is not available. Prioritize the 5 choices, and email Margi the list. We will do our best to satisfy your preferences.

Collaborative Modeling Sessions Timetable

All times are in Pacific Time (LA Time)

From	To	3-22	3-23	3-24	3-25	3-26	3-27	3-28	3-29	3-30
9 AM	11 AM	J	J		J	J	J	J	J	J
11 AM	1 PM									
1 PM	3 PM	J	J	J	J	J		J	J	J
3 PM	5 PM	J	J	J	J	J	J	J	J	J
5 PM	7 PM									
7 PM	9 PM	J	J	J	J	J	J	J	J	J

Task Details and Deliverables

There will be a separate document that describes the tasks and deliverables for Part 2. It will be disseminated soon.

Timeline

By noon on	Action Items
March 7th	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
March 11th	Email us 5 available times (with priority) for your collaborative session
March 22nd – 30th	Collaborative design and modeling sessions
March 31st	HW3 due

Filename Conventions

You should submit four files (one .mga file from part 1, two .mga files from part 2, and a .pdf file that has all required discussions from both part 1 & 2) in a single zipped file.

File	Filename Format
Submission Package	• *<lastname>_<firstname>_csci578_mattmann_spring2014_HW3.zip*
Part 1 Model	• *<lastname>_<firstname>_csci578_mattmann_spring2014_HW3_m1.mga*
Part 2 Models	With proactive conflict detection: • *<lastname>_<firstname>_csci578_mattmann_spring2014_HW3_m2_w.mga* Without proactive conflict detection: • *<lastname>_<firstname>_csci578_mattmann_spring2014_HW3_m2_wo.mga*
Discussion	• *<lastname>_<firstname>_csci578_mattmann_spring2014_HW3.pdf*

Questions

Should you have further questions, send an email to the TA (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] Jae young Band, Ivo Krka, and Nenad Medvidovic. "How software architects collaborate: Insights from collaborative software design in practice." In proceedings of 6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE), 2013. IEEE, 2013.

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

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

[6] 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.

[7] 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.

[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.

[11] Yuriy Brun, Reid Holmes, Michael Ernst, and David Notkin. “Early Detection of Collaboration Conflicts and Risks.” IEEE Transactions on Software Engineering, 2013.