Project Overview

In teams of three or four students, you will design, implement, and document a release pipeline capable of achieving "continuous delivery" of software releases. You will deliver the system in iterations throughout the term. Several deliverables are to be submitted at checkpoints throughout the term as described below.

1 High-Level Description of the Subject System

TouchCORE is a multitouch-enabled tool for agile concern-oriented software design modeling aimed at developing scalable and reusable software design models. Currently, the tool gives the designer access to a vast library of reusable models encoding essential recurring design concerns (security, fault tolerance, distribution, design patterns). It exploits model interfaces and aspect-oriented model weaving techniques as defined by the Reusable Aspect Models (RAM) approach to enable the designer to rapidly apply reusable design concerns within the design model of the software under development.

TouchCORE is built on top of the Eclipse Modeling Framework (EMF) and MT4J (multi-touch library for Java). The current version has been tested on or 32 and 64 bit architectures on Mac OSX (Java 1.8 or newer), Windows and Linux (Ubuntu) platforms.

2 The Pipeline

The project outcome (a release and deployment pipeline) will provide several features for the TouchCORE development team (members of ECSE and SoCS at McGill University). As an end-to-end solution, the pipeline will automate the integration, build, and deployment routines. More specifically, the project is broken down into a series of deliverables, which are described below. More detail about the specific deliverables will be announced later.

2.1 Build System

At the core of every release pipeline is a build system. Project need a build system to automate the process of (re)synchronizing source code with the set of intermediate files and deliverables. The build system of TouchCORE should support both the incremental building as well as the automated execution of automated tests.

2.2 Continuous Integration

For each proposed patch, a continuous integration process will be initiated to automatically create a build job. Each build job will include quality gates that check that the patch does not introduce (1) compilation errors, (2) test regressions, (3) static analysis warnings, and (4) style check issues. The CI process will need to be integrated with the pull request-based integration strategy of the project on BitBucket.

2.3 Continuous Delivery

For proposed patches that satisfy continuous integration and reviewing criteria, the development team will be presented with the option to immediately create and publish a release.

3 Scope and Technology Constraints

Your pipeline must support the scenarios described in Section 2. For specific technology choices that you use to implement the various pipeline phases, you are free to choose any technology that the representatives of the TouchCORE team will approve. We will have consultation meetings with the team throughout the semester, so please clear your choices up with them during those meetings.