

Jeffrey Goett
August 21, 2003

Using a Docking Process to Ensure a Clean Software Migration

Abstract

The Notre Dame Open Source Software group used a docking procedure to help ensure a clean migration of their simulations from Swarm to Repast. These computer simulations model four different mechanisms driving the growth of the “social” networks at Sourceforge.com. The docking process narrowed in discrepancies between the Swarm simulation and the corresponding Repast migration, beginning with a comparison of macroscopic output characteristics of corresponding models but eventually moving onto a comparison of the microscopic actions of each time step. This process found a difference between the schedule implementations of Swarm and Repast and a problem with the setup of the SQL database. After correcting these problems, the Swarm simulation and the Repast migration successfully docked.

I. Background

The Notre Dame Open Source Software group simulates with computer programs four different mechanisms driving the growth of the social network at sourceforge.com. With the development of Repast, these programs were migrated from Swarm to Repast. A docking process helped to make this migration as clean as possible.

Each of the four simulations uses a different model to simulate the growth of the topology of sourceforge’s social network. One simulation uses the random link reconnection model proposed by Paul Erdős and Alfréd Rényi. The three other models use the notion of preferential attachment suggested by Albert-Laszlo Barabasi and Réka Albert. A comparison with empirical data from Sourceforge gauges the accuracy of each model.

The community at Sourceforge.com can be viewed as a social network. In the spirit of open source software development, any software developer can join an ongoing software project at the site. This can be modeled as a graph: one type of node represents a project, another a developer. Edges exist between a developer and the project(s) he or she works on.

This network topology, however, does not clearly depict the “social” relationships present in the network that interest the Notre Dame group. Two different “views” of this network show these two social relationships. In the first view, developers are “socially” connected to each other if they work on the same project. These many relationships are illustrated graphically: nodes represent the developers and edges represent the links between developers. In the second view, projects are “socially” (in a loose sense) connected to one another if they share a common developer. Graphically, nodes represent each projects while edges represent links between the projects sharing at least one developer. The Notre Dame group studies the topological properties of these two graphs. They compare the degree distribution, the number of nodes, and the diameter from the simulated data with that from the empirical data.

All four simulations model the acts of developers joining and abandoning projects. Each simulation, however, varies slightly in the method the developer uses in choosing which project to join. The ER model uses a random selection method suggested by Paul Erdős and Alfréd Rényi. The BA, BA constant fitness, and BA dynamic fitness

models all use variations on the concept of preferential attachment suggested by Albert-Laszlo Barabasi and Réka Albert .

The improvements offered by Repast over Swarm prompted the migration of these four simulations to Repast. A docking procedure helped maintain the cleanliness of this migration by comparing the output of corresponding models in Swarm and Repast.

II. Procedure

This docking process found a difference between the schedule implementations of Swarm and Repast and a problem with the setup of the database. To do so, the docking process narrowed in on these problems, beginning with comparison of macroscopic output characteristics between corresponding models but eventually moving onto a comparison of the microscopic actions of the simulation.

In this comparison of Swarm and Repast, the Notre Dame group ran simulations of the BA model without fitness.

The first attempt at docking compared the macroscopic characteristics of degree distribution and size vs. time between the two corresponding BA without fitness models. The Swarm simulation used Swarm's built in random number generator. The Repast simulation used the colt random number generator from CERN. A graphical comparison of degree distribution for projects and developers over multiple runs of Swarm and Repast revealed systematic differences between the two packages. Over one subdomain of the developer degree distribution, Swarm had a higher count than Repast. Over another subdomain, Swarm had a lower count. The next step in the process would determine whether or not the random number generators caused this systematic difference.

To determine this, the group ran the two simulations using the exact same set of random numbers: each simulation used the same random number generator with the same seed. The developer degree distribution data from these runs, however, revealed similar systematic differences between the two simulations.

To determine the exact reasons for this difference, the group had the simulations log the action that each developer took during each step. Comparing these logs, two reasons for the differences emerged.

First, the group discovered that one simulation would occasionally throw an SQL Exception. To recover from such an error, the simulation ignores the developer's action and moves on to the next developer. Since the developer's previous actions affect its future actions, this error can cause more discrepancies at future time steps. One group member, Yongqin Gao, found the cause of this error to be a problem with the primary keys in the links table of the database.

Further inspection of the data logs showed that Swarm snapshots taken during the simulation were out of phase by one time unit with the snapshots of Repast. Even if each simulation's developers acted identically, this extra time step prevented the snapshots of data from matching. The Swarm scheduler begins at time step 0 while the Repast scheduler begins with time 1. For snapshots originally thought to be corresponding, Swarm had actually executed one more time step than Repast. Since the output comparisons used these data snapshots, all comparisons were invalid.

With these two problems corrected, the corresponding logs of developers' actions matched. Using the same sequence of random numbers, the Swarm and Repast simulations of the BA model produced identical output.

III. Conclusion

The docking process described above helped to reveal differences between the Swarm and Repast schedulers. Additionally, it found a problem with the setup of the "links" table in the SQL database. Such a docking process proved to be crucial both in creating a clean migration and in understanding better the simulations and the Swarm and Repast software packages.