Sequential Integration

Without controlling integration developers test their code and integrate believing all is well. But because of parallel integration there is a combination of source code which has not been tested together before. Integration problems happen without detection.

Further problems happen when there is no clear cut latest version. This applies not only to the source code but the unit test suite which must verify the source code correctness. If you can not lay your hands on a complete, correct, and consistent test suite you will be chasing bugs that do not exist and passing up bugs that do.

You can have developers own specific classes. Class owners integrate and commit to the repository properly. Interclass dependencies can still be wrong. Class owners can become bottle necks if one class requires many changes and it doesn't solve the whole problem.

Yet another way is to appoint an integrator or integration team. Integrating code from multiple developers is more than a single person can handle. And a team of people is too big a resource to integrate more than once a month. In this environment developers work with obsolete versions which are then erroneously re-integrated into the code base.

Many projects use on a source code repository tool to control integration. Most source code repositories seem to encourage sequential development and parallel integration. That is, they lock a file so only one developer can add functionality at a time. They allow anyone to release at anytime because overwrites have been controlled at the individual file level.

These solutions do not address the root problem. You want developers to be able to proceed in parallel, making changes to any part of the system required, but you also want an error free integration that doesn't lose changes. Like a dozen steaming locomotives headed for the switching yard at the same time, there is going to be trouble. Instead of restricting development to being sequential in nature let's rethink the problem. Our locomotives can all get into the switching yard without a crash if they just take turns. We need to do this with code integration as well.

Strictly sequential (or single threaded) integration by the developers themselves in combination with collective code ownership is a simple solution to this problem. All new code is released to the source code repository by taking turns. That is, only one development pair integrates, tests and commits changes at any given moment. Single threaded integration allows a latest version to be consistently identified.

This is not to imply that you can not integrate your own changes with the latest version at your own workstation any time you want. You just can't release your changes to the team without waiting for your turn.

Development proceeds in parallel while integration is sequential. It seems counter intuitive to many developers because their experience is that integration takes forever so it should be done as few times as possible. Instead integrate more often so that it is very fast and easy.

Some sort of sequencing mechanism is required. The simplest thing is a single computer dedicated to integration. Development pairs are implicitly sequenced by take turns using it. But a physical token passed from developer to developer also works. Integrating and releasing code often shortens the time needed to hold the lock and thus minimizes the wait time to acquire the lock.