# Jetstream Code Management

Howard Buckley
January 23, 2012

# Background

- Jetstream is an integrated set of computational tools for the purpose of 3D aerodynamic shape optimization
- Hicken wrote the original version of Jetstream as part of his doctoral research (c. 2006 - 2009)
- The group has continued to develop Jetstream. Each member adding functionality associated with their thesis research
- Development has proceeded in an unorganized way resulting in the code splintering into several divergent, irreconcilable versions.
- Recently several group members have worked hard to put the pieces back together into one coherent version of Jetstream
- Question: How do we prevent this from happening again?!
- Answer:  Some management of code development is required

# Why is this important?

- The whole is greater than the sum of its parts; i.e.
  - The usefulness of the code as a tool arises from the integration of all its parts
- Your contributions are at risk of being forgotten / irretrievable / lost etc... if your code is not in sync with the rest of the group
- Accelerated progress when all members are moving in the same direction
- Reduces likelihood of members 're-inventing the wheel'
- Greater chance of catching bugs early when they might be easier to resolve

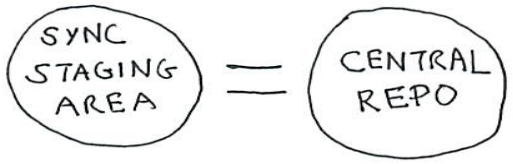
# Jetstream Code Sync Procedure

The following is a procedure for synchronizing all group member's versions of Jetstream. To be effective it must be performed on a regular basis, for example every month.

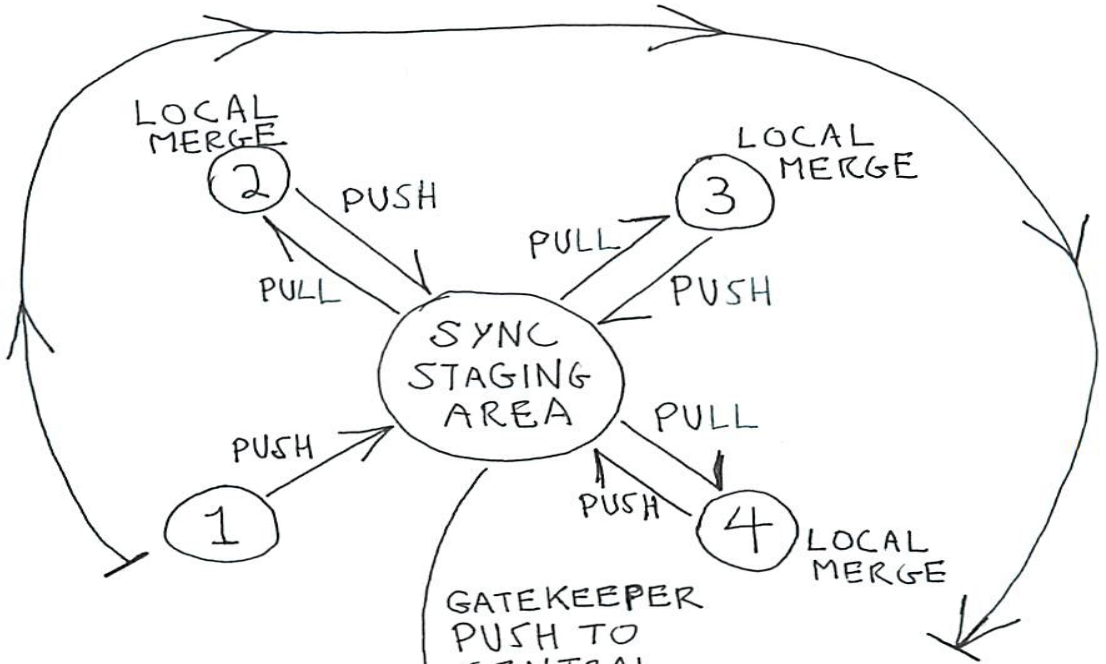
1. Initialize synchronization staging area:
    - Gatekeeper to clone current baseline version of Jetstream from central repository
2. Merge:
    - A group of 3 or 4 members are required to merge their latest changes into sync staging area.
3. Baseline Update:
    - After merging, final commit to sync staging area is pushed to the central repo by the gatekeeper
4. Group Synchronize:
    - All members synchronize with the updated baseline code in the central repository
5. Group Testing:
    - All members of the group run their assigned test case(s), debug as required, and update test case status

# JETSTREAM CODE SYNCHRONIZATION PROCESS

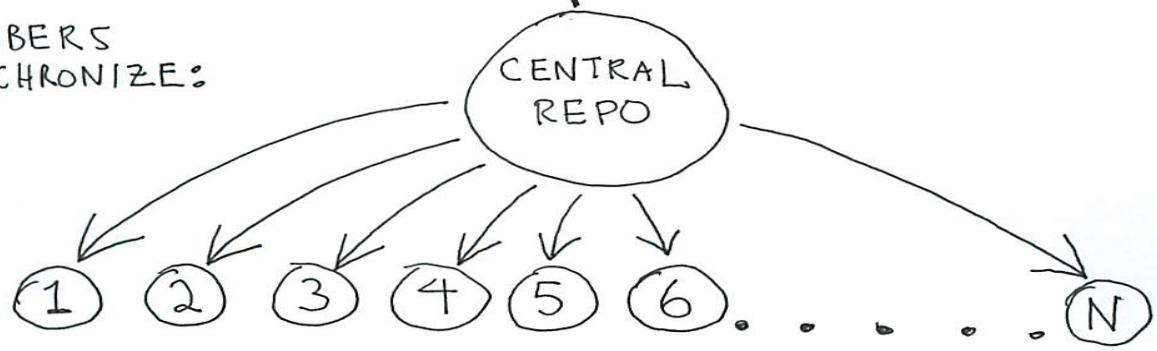
IN THE BEGINNING...




MERGING:
(4 AT A TIME)




ALL MEMBERS SYNCHRONIZE:



# Merge Sub-Groups

- Small groups to keep the merge process manageable
- Cycle through groups on a monthly basis
- Allows individuals 4 months of development time between merges

**Group 1:**

Michal Osusky
Lana Osusky
Tim Leung
Tom Reist

**Group 2:**

Xiaodong Wang
David Boom
David Del Rey Fernandez
Kwesi Apponsah

**Group 3:**

Karla Telidetzki
Howard Buckley
Hugo Gagnon
David Brown

**Group 4:**

Jenmy Zhang
Abdalla Elraghy
Ramy Rashad
Anh Truong

# Group Testing for Code Verification (Did we break anything?)

- Divide and conquer! Split up the task of code testing to reduce time
- Members have ownership of one or more test cases
- Ideally test case(s) are matched with members' area of research
- Test cases are part of the Jetstream Test Suite
- What steps are involved?
  1. Run the test case
  2. Check results against a handful of benchmark values
  3. Update test case status on Jetstream Test Suite Summary Google Doc (PASS or FAIL)
  4. Debug as required
- Will require communication and cooperation amongst group members
- Jetstream Test Suite link: http://bit.ly/nuqUyI

# Good Coding Practice

- Work with at least 2 GIT branches e.g. master and development
- GIT commit often
- Control code configuration with input parameters
  - allows the code to run with combinations of features and user-specified parameter values relevant for a given case
- Avoid hard-coding values or chunks of code with pre-processor directives
  - It's a pain to re-compile everytime you need to change a hard-coded value, etc..
- Add new subroutines to the bottom of existing module files
  - Reduces merging headaches
- Link to grid files instead of creating copies in run directories
  - ln -s /project/z/zingg/grid_library/021/grid_021.g  grid_021.g
- See if a suitable grid exists in Grid Library before creating a new one from scratch

# Grid Library

- A repository of grids available to all group members:
  - on Scinet: /project/z/zingg/grid_library
- May contain both 2D and 3D grids
- Each grid is stored in a numbered folder, e.g.
  - grid_library/017
  - folder contains at least a grid file and connectivity file
  - grid files are named according to their folder number, e.g.
    - grid_017.g, grid_017.con
- Ideally the grid generation files are also included in the folder. e.g. ICEM blocking, ICEM scripts
- Grids are described in the Grid Library Summary Spreadsheet
  - Link to Grid Library: http://bit.ly/rBx2hi
- Let others benefit from your hard work...consider sharing your grids (and the ICEM files used to generate them) with the group

# Questions?