Best Practices for Software Development in the Research Environment

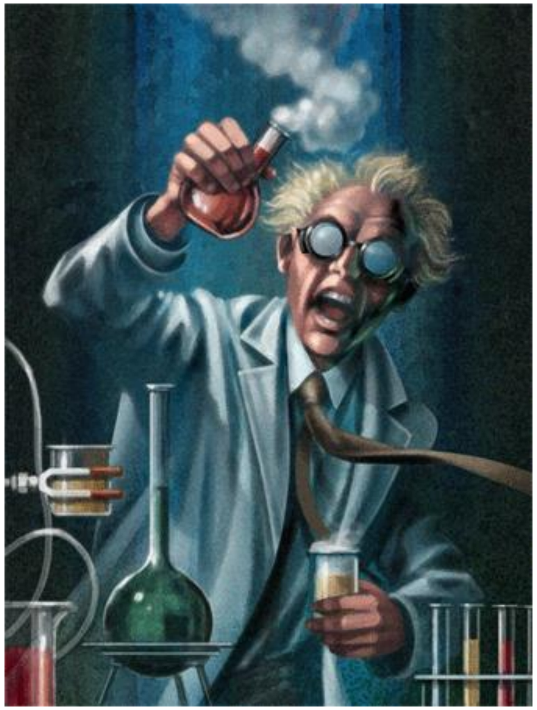
Adam Brazier – *brazier@cornell.edu*
Computational Scientist
Cornell University Center for Advanced Computing (CAC)

Why?

- We know how to write computer programs! You can't teach us anything!
  - None of this is *compulsory*.
  - Many of the topics are low-hanging fruit, with relatively obvious advantages; they also link together and support each other
  - Knowing what's out there is a big advantage even if you don't use it
    - Can make transitions when they make sense for your working practices
    - Can understand the software others who are using these practices
  - HPC produces good output quickly, but it can also produce bad output – and cost a lot of money – very quickly as well. The stakes are high.

The research environment is different from many other software environments

- HPC a key focus
- Time limitations
  - Opportunity cost big concern, c.f. financial
- Target audience are expert
- The software is not the end goal in itself
- Coders often from research domains
- Individual “ownership” of software products common


Mad? You call me mad?

Overview

- Coding practices
  - (because most of us are already coding)
- Software Lifecycle
  - Cradle to Grave
- Improving the Software
  - Faster. Better. Stronger

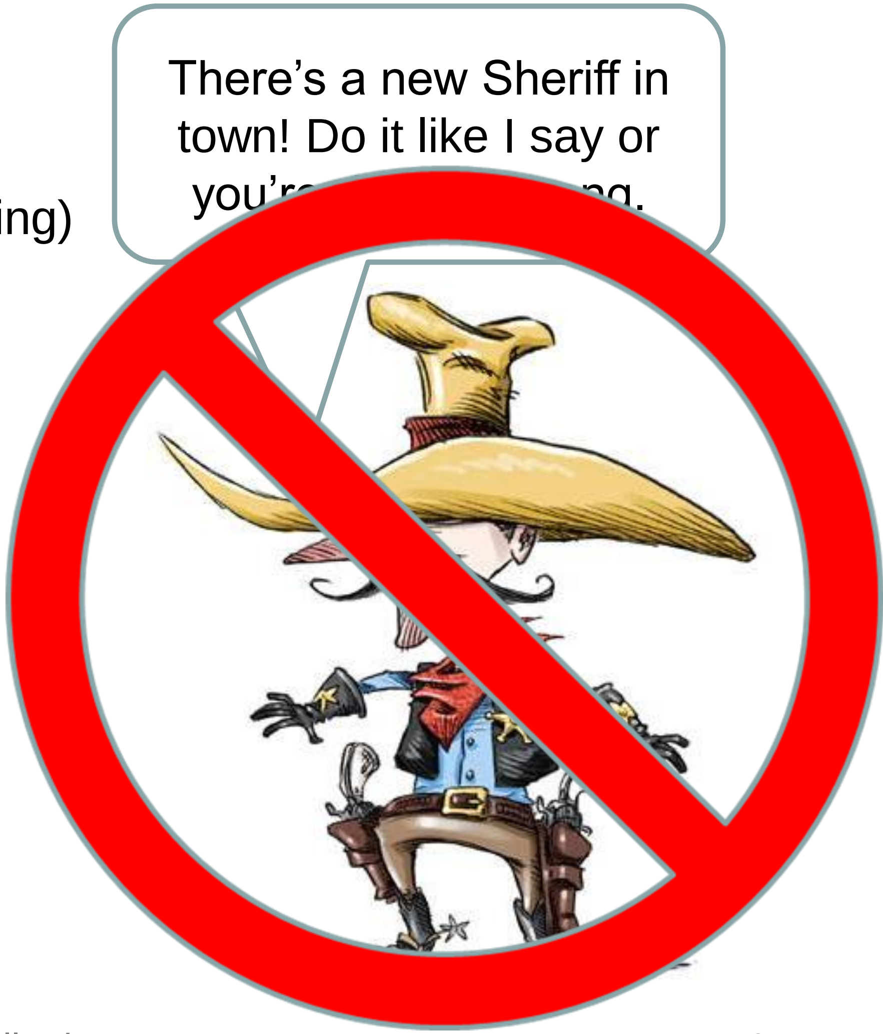
Overview

- Coding practices
  - (because most of us are already coding)
- Software Lifecycle
  - Cradle to Grave
- Improving the code
  - Faster. Better. Stronger

There's a new Sheriff in town! Do it like I say or you're doing it wrong.

Overview

- Coding practices
  - (because most of us are already coding)
- Software Lifecycle
  - Cradle to Grave
- Improving the code
  - Faster. Better. Stronger


There's a new Sheriff in town! Do it like I say or you'r ng.

Who needs what?

Group	Solo coders	Team coders	Project manager	User
Code clarity, commenting	✓✓	✓✓✓	✓	✓
Documentation	✓	✓	✓✓	✓✓✓
Versioning	✓✓✓	✓✓✓	✓✓✓	✓
Requirements	✓✓	✓✓	✓✓✓	✓✓✓
Estimation	✓✓	✓✓	✓✓✓	✓
Design	✓✓	✓✓	✓	✓
Testing	✓	✓✓	✓	✓
Deployment, upgrades, integration	✓	✓✓	✓✓	✓✓

Overview

- Coding practices
  - (because many of us are already coding)
- Software Lifecycle
  - Cradle to Grave
- Improving the Software
  - Faster. Better. Stronger

Coding Practices

- Readability of the code
- Code organization
- Commenting and documentation
- Using an IDE
- Versioning and source control

The Rosetta Stone. You shouldn't need one of these to follow code

Coding practices: a general rule to remember

Coding is a human endeavor!

- Humans can absorb information, see patterns and comprehend structure very well if the information is formatted appropriately
- People forget things; documentation is important
- Much of the work you do on your code is not the initial writing. Appropriate effort making it understandable is repaid again and again.
- Even machine-generated code may have to be read by humans.

Coding Practices: Readability

- There is no shame in writing code to be readable. The opposite, in fact!

- Targets:
  - Future you
  - Other people

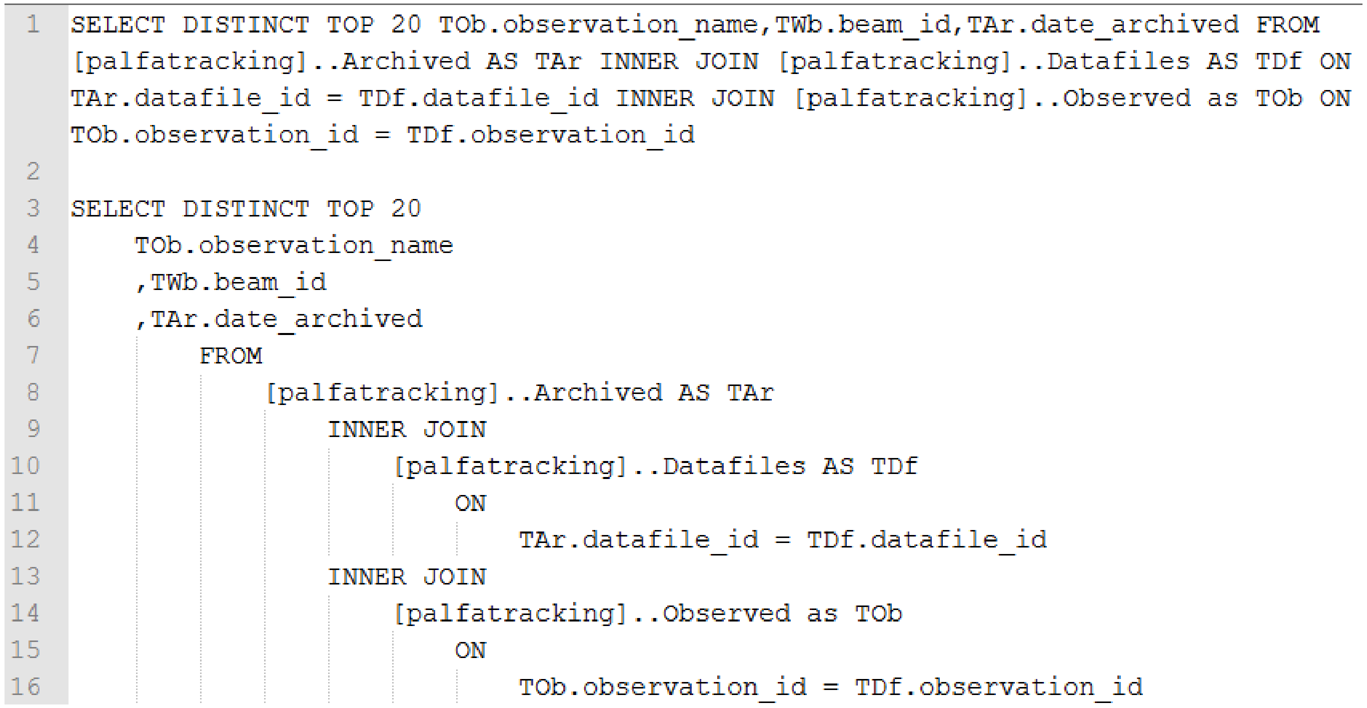
Dogs prefer readable code, too!

- "Readability" means being clear to *people*
  - People are good at recognizing patterns
  - People do not generally like dense text

- Most languages have their own style preferences

Coding Practices: simple readability example: formatting

- Both of these execute the same (taken from a longer code)



```
SELECT DISTINCT TOP 20 TOb.observation_name,TWb.beam_id,TAr.date_archived FROM [palfatracking]..Archived AS TAr INNER JOIN [palfatracking]..Datafiles AS TDf ON TAr.datafile_id = TDf.datafile_id INNER JOIN [palfatracking]..Observed as TOb ON TOb.observation_id = TDf.observation_id

SELECT DISTINCT TOP 20
    TOb.observation_name
    ,TWb.beam_id
    ,TAr.date_archived
        FROM
            [palfatracking]..Archived AS TAr
                INNER JOIN
                    [palfatracking]..Datafiles AS TDf
                        ON
                            TAr.datafile_id = TDf.datafile_id
                INNER JOIN
                    [palfatracking]..Observed as TOb
                        ON
                            TOb.observation_id = TDf.observation_id
```

Coding Practices: some general readability rules

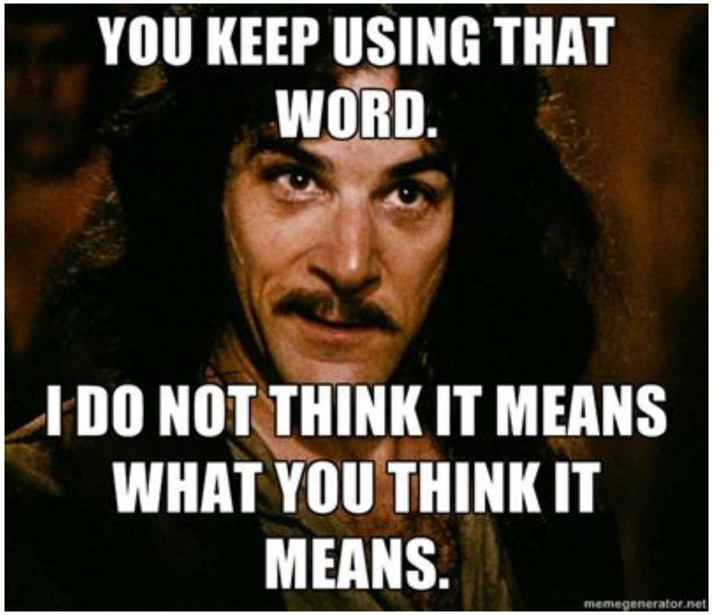
- Use whitespace to:
  - Group code together (particularly by indentation)
  - Make code more readable
  - Spaces instead of tabs (IDE can be set to insert spaces when you tab)

- Do not produce very long lines of code.
  - Shorter lines allow for multiple editor windows
  - 80 or 100 characters a good limit

- Limit the number of text lines in a given file if possible
  - But don’t try to pack a lot of operations into one line!
    - Unless performance benefits (in which case, comment)

Coding Practices: semantic content

- Names can be informational
  - Variable names describe content and/or type
  - Method names describe function
  - File names summarize contents
  - Aids self-documentation of code
- Be consistent
- Avoid ambiguity
- Spend time on this!
  - Check out Ottinger's rules


YOU KEEP USING THAT WORD.
I DO NOT THINK IT MEANS WHAT YOU THINK IT MEANS.
memegenerator.net

Misunderstandings are bad

Coding Practices: organization of code

- Use multiple files, appropriately named
  - Can't find a suitable name for a file to summarize its contents?
    - Perhaps there should more than one file
  - Makes code more manageable
  - Separate data from code
    - No magic numbers or strings!
  - Allows simpler collaborative coding
  - Enhances reusability
- Use multiple directories
  - Software often has subproducts
  - Datafiles stored away from code
  - Namespaces (see next)

Organization is good

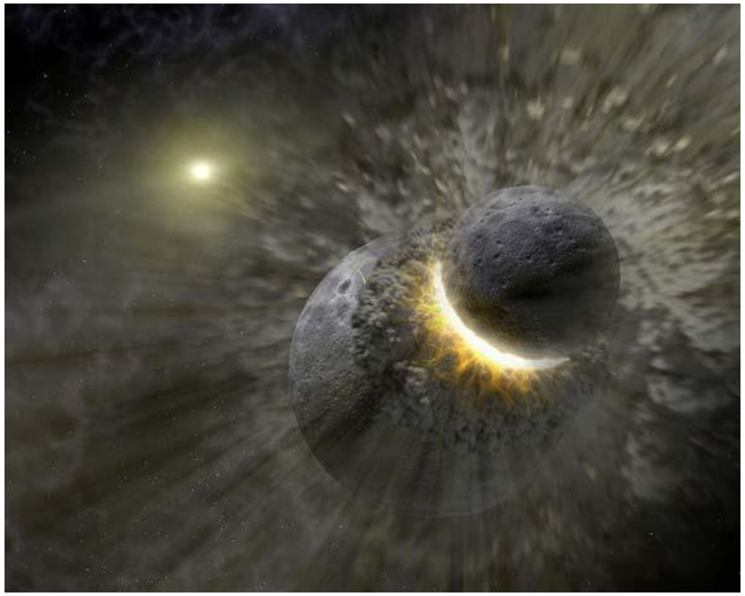
Coding Practices: namespacing

- Why?
  - Avoids collisions
    - Enables descriptive naming
    - Simpler integration of software

```
os.path.join(a, *p)
forests.path.join(path1, path2)
```

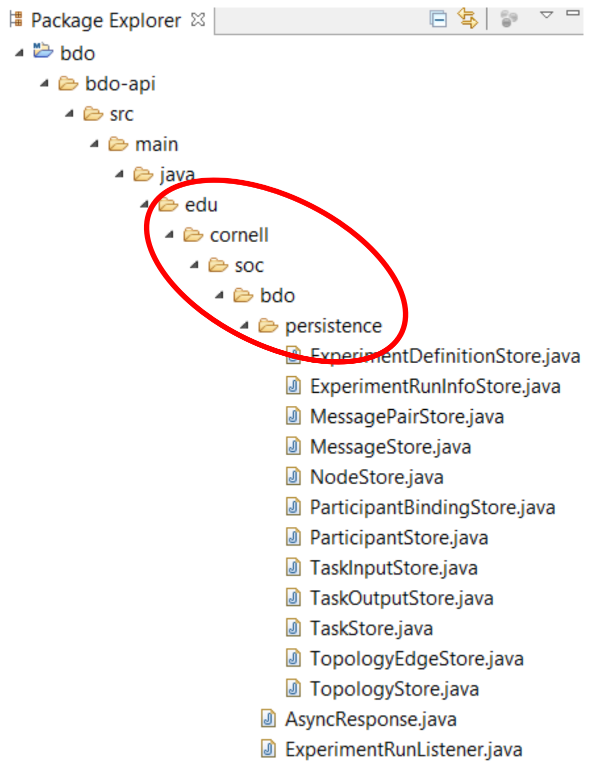
- Typically mirrors directory structure

- Plan it ahead of time
  - Part of design process


Collisions are bad

Example: the `edu.cornell.soc.bdo.persistence` namespace

- `Functions are, eg, edu.cornell.soc.bdo.persistence.taskStore.<function>`



```
Package Explorer
bdo
  bdo-api
    src
      main
        java
          edu
            cornell
              soc
                bdo
                  persistence
                    ExperimentDefinitionStore.java
                    ExperimentRunInfoStore.java
                    MessagePairStore.java
                    MessageStore.java
                    NodeStore.java
                    ParticipantBindingStore.java
                    ParticipantStore.java
                    TaskInputStore.java
                    TaskOutputStore.java
                    TaskStore.java
                    TopologyEdgeStore.java
                    TopologyStore.java
                  AsyncResponse.java
                  ExperimentRunListener.java
```

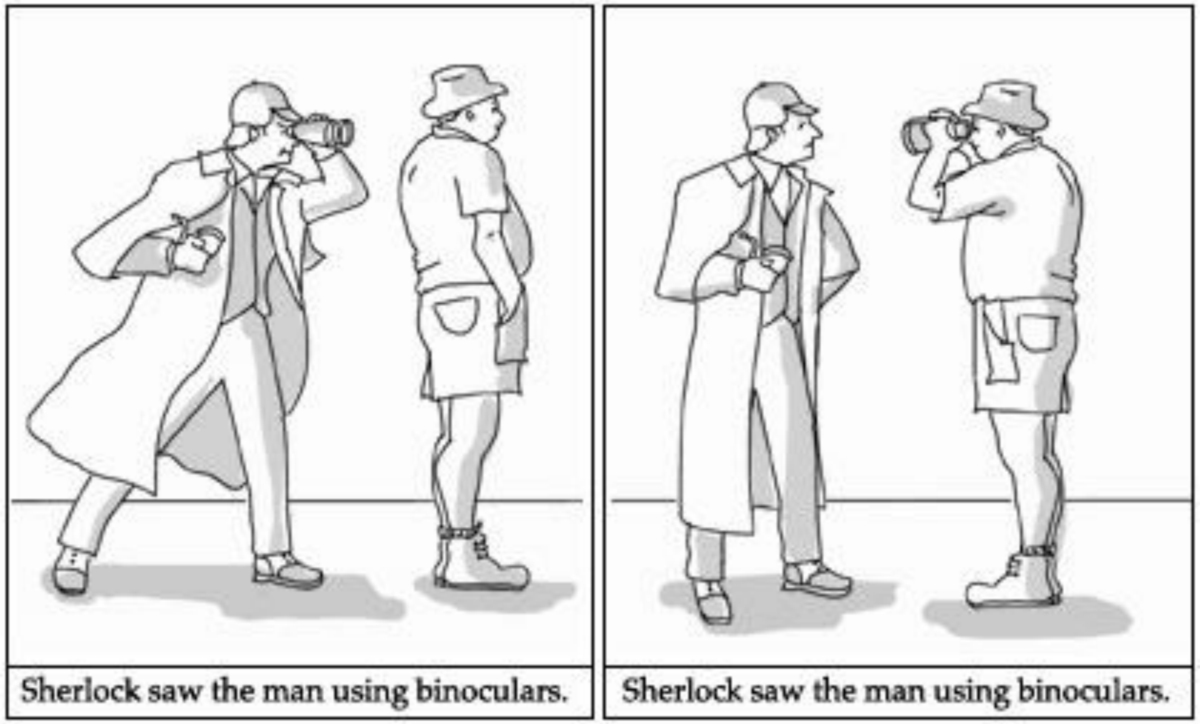
Coding Practices: commenting and documenting

- Typically:
  - Documentation explains what software product does, written for users
  - Comments refer to the code itself, written for coders
  - In the research environment, users and coders are often the same

- Comments and documentation are free-form
  - Can be written purely to communicate information
  - Provide substantial help to their target audience
  - Enhance productivity and mitigate problems:
    - Comments are not a substitute for good coding, but do make the code easier to understand
    - Documentation not a substitute for deficiencies in the software but can enhance user productivity and/or reduce user frustration

Coding Practices: Comment to prevent problems

- Comments should make clear what might otherwise not be clear


Sherlock saw the man using binoculars.
Sherlock saw the man using binoculars.

When you have done what you can to eliminate the impossible, there may still be more than one answer

Coding Practices: useful comments

- *May* summarize code functionality
  - self-documenting code better

- May explain things which might not be obvious at first glance

- May explain *why* things are done the way they are

- May highlight future work
  - e.g., IDE may collate `TODO` elements to produce a punchlist for future work

Coding practices: some commenting examples

Useful

```
#Removing colons from file name
#in case this is run on windows
filename = filename.replace(':', '-')
```

```
#Database can't handle unicode
name = name.encode('ascii','ignore')
```

```
#returns tuple for getitem rather than
#list, to duplicate pyodbc row behavior
```

```
#TODO: add datetime (which must also
#be quoted like a string)
```

Not-good

```
#Don't need this anymore
```

```
#Need to fix this!
```

```
#Function returns int
```

```
#Iterates through list of animals
#and selects quadrupeds. Lister must
#be a list of animals
def finder(lister):
```

Coding practices: some commenting examples

So delete it (and the comment!)

Fix what? What's wrong?

Make it obvious in the code!

Pick better names! E.g.,

```
def quadruped_finder (animal_list):
```

```
string[] quadruped_finder(Animals
animalList){
}
```

<u>Not-good</u>

```
#Don't need this anymore
```

```
#Need to fix this!
```

```
#Function returns int
```

```
#Iterates through list of animals
#and selects quadrupeds. Lister must
#be a list of animals
def finder(lister):
```

Coding Practices: and…

- As comments aren’t executed, they may be used in debugging and development to exclude code sections without deleting them
- Don't leave commented-out code in production version!
- On longer timescales, use versioning

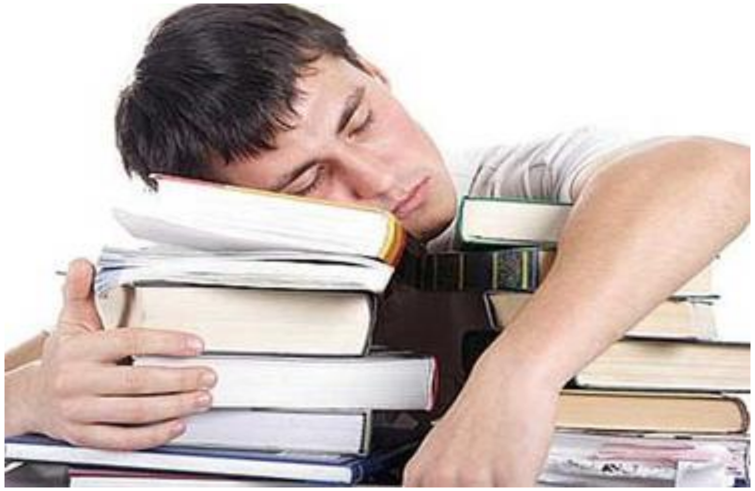
Coder at work

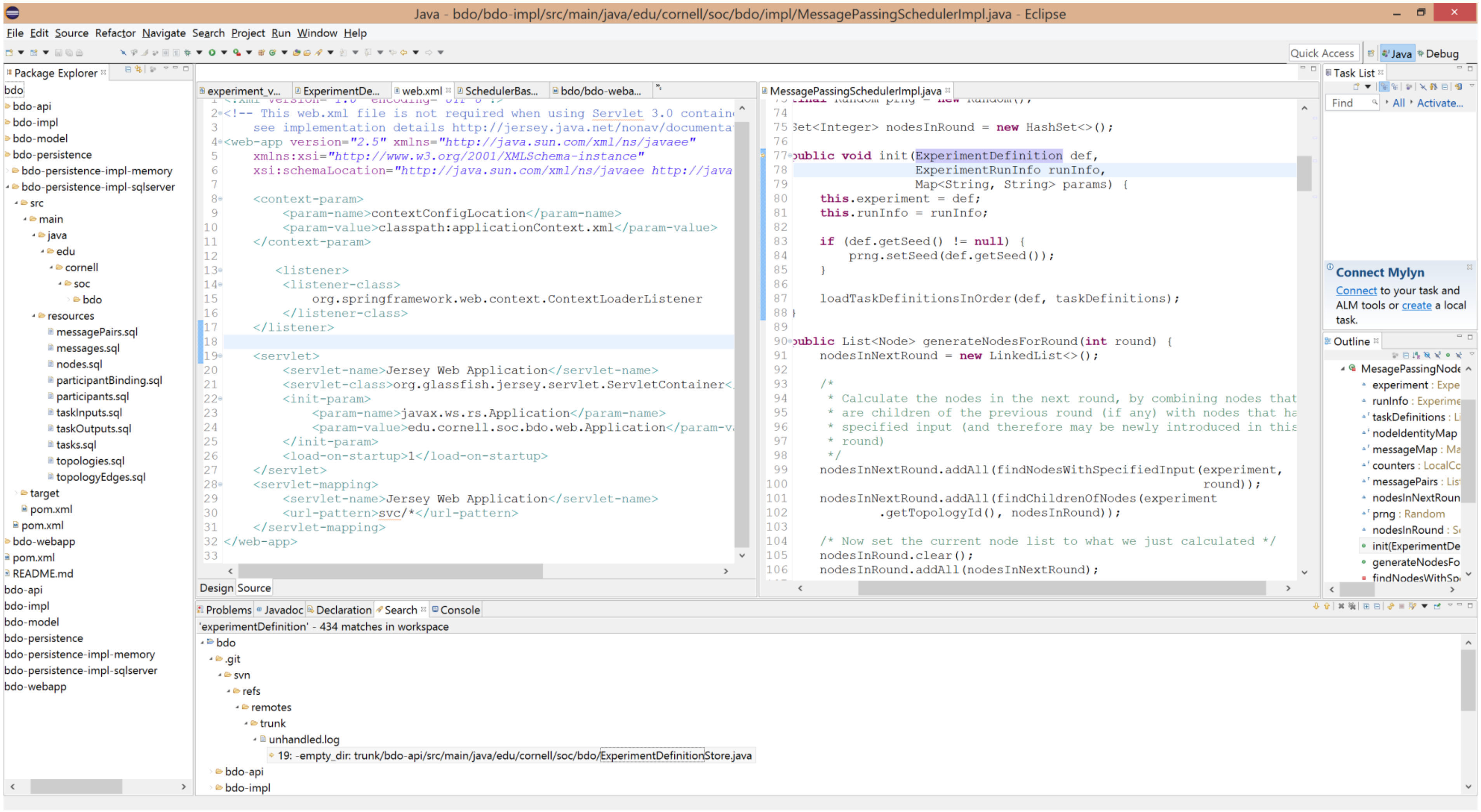
Coding Practices: documentation

- Its value typically increases as time passes
- It is typically produced at the end

$\rightarrow$ Easily neglected

- To reduce the impact of this neglect:
  - Self-documenting code
  - Document generation:
    - May use commenting features
    - May be code-specific (e.g., javadocs, python docstrings, etc)
    - Can be achieved with other software (e.g., Doxygen, Sphinx).


Coding's done. Can I sleep now?


Java - bdo/bdo-impl/src/main/java/edu/cornell/soc/bdo/impl/MessagePassingSchedulerImpl.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
Debug
Package Explorer
experiment_v...
ExperimentDe...
web.xml
SchedulerBas...
bdo/bdo-weba...
MessagePassingSchedulerImpl.java
Task List
Find
All
Activate...
Connect Mylyn
Connect to your task and ALM tools or create a local task.
Outline
Design
Source
Problems
Javadoc
Declaration
Search
Console

```
bdo
  bdo-api
  bdo-impl
  bdo-model
  bdo-persistence
  bdo-persistence-impl-memory
  bdo-persistence-impl-sqlserver
    src
      main
        java
          edu
            cornell
              soc
                bdo
        resources
          messagePairs.sql
          messages.sql
          nodes.sql
          participantBinding.sql
          participants.sql
          taskInputs.sql
          taskOutputs.sql
          tasks.sql
          topologies.sql
          topologyEdges.sql
    target
    pom.xml
  pom.xml
  bdo-webapp
  pom.xml
  README.md
bdo-api
bdo-impl
bdo-model
bdo-persistence
bdo-persistence-impl-memory
bdo-persistence-impl-sqlserver
bdo-webapp
```

```
<!-- This web.xml file is not required when using Servlet 3.0 contain
    see implementation details http://jersey.java.net/nonav/documenta
<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java

    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>classpath:applicationContext.xml</param-value>
    </context-param>

       <listener>
        <listener-class>
            org.springframework.web.context.ContextLoaderListener
        </listener-class>
    </listener>

    <servlet>
        <servlet-name>Jersey Web Application</servlet-name>
        <servlet-class>org.glassfish.jersey.servlet.ServletContainer<
        <init-param>
            <param-name>javax.ws.rs.Application</param-name>
            <param-value>edu.cornell.soc.bdo.web.Application</param-v
        </init-param>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>Jersey Web Application</servlet-name>
        <url-pattern>svc/*</url-pattern>
    </servlet-mapping>
</web-app>
```

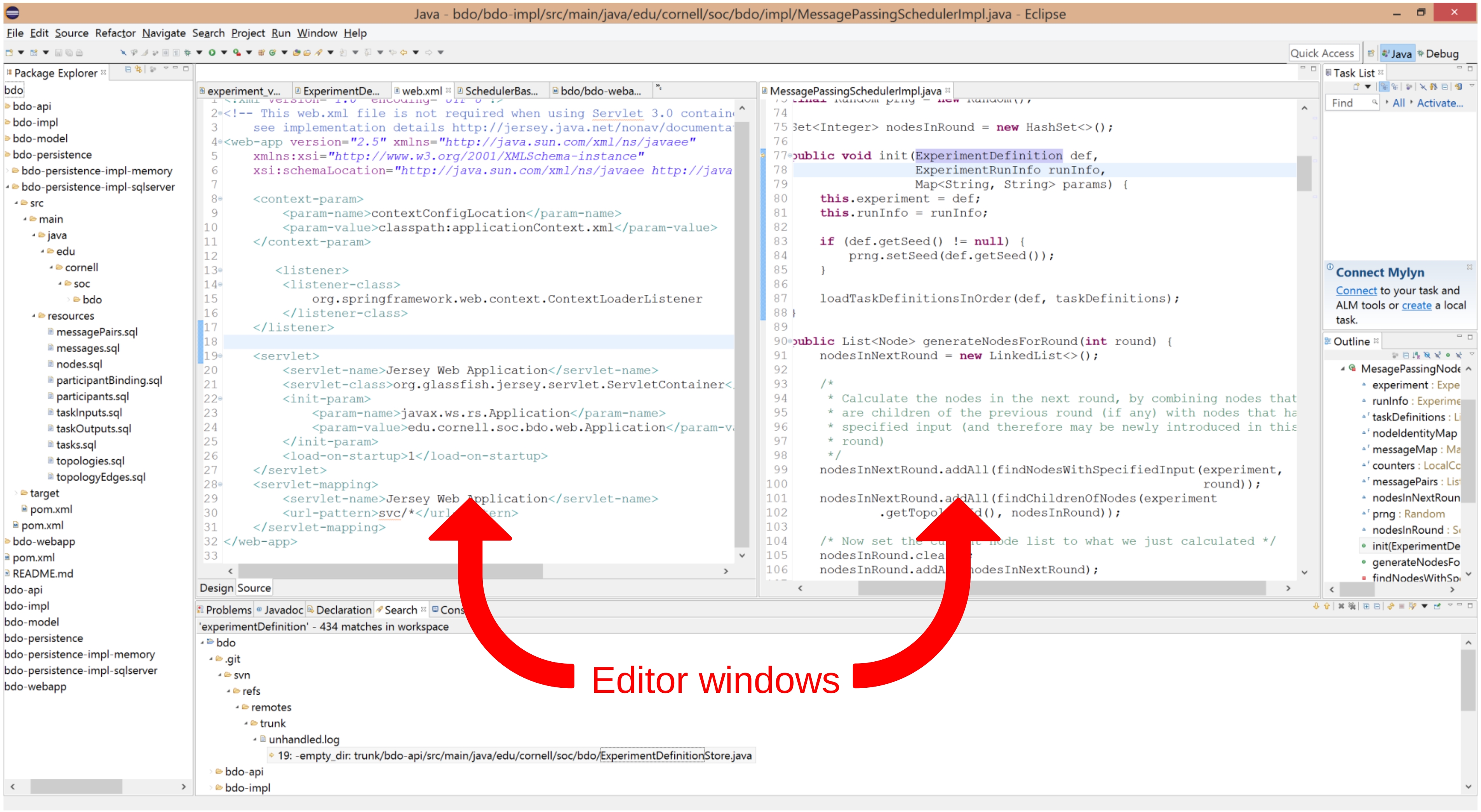
```java

Set<Integer> nodesInRound = new HashSet<>();

public void init(ExperimentDefinition def,
                 ExperimentRunInfo runInfo,
                 Map<String, String> params) {
    this.experiment = def;
    this.runInfo = runInfo;

    if (def.getSeed() != null) {
        prng.setSeed(def.getSeed());
    }

    loadTaskDefinitionsInOrder(def, taskDefinitions);
}

public List<Node> generateNodesForRound(int round) {
    nodesInNextRound = new LinkedList<>();

    /*
     * Calculate the nodes in the next round, by combining nodes that
     * are children of the previous round (if any) with nodes that ha
     * specified input (and therefore may be newly introduced in this
     * round)
     */
    nodesInNextRound.addAll(findNodesWithSpecifiedInput(experiment,
                                                        round));

    nodesInNextRound.addAll(findChildrenOfNodes(experiment
            .getTopologyId(), nodesInRound));

    /* Now set the current node list to what we just calculated */
    nodesInRound.clear();
    nodesInRound.addAll(nodesInNextRound);
```

```
MessagePassingNode
  experiment : Expe
  runInfo : Experime
  taskDefinitions : Li
  nodeIdentityMap
  messageMap : Ma
  counters : LocalCo
  messagePairs : Lis
  nodesInNextRoun
  prng : Random
  nodesInRound : S
  init(ExperimentDe
  generateNodesFo
  findNodesWithSp
```

```
'experimentDefinition' - 434 matches in workspace
bdo
  .git
    svn
      refs
        remotes
          trunk
            unhandled.log
              19: -empty_dir: trunk/bdo-api/src/main/java/edu/cornell/soc/bdo/ExperimentDefinitionStore.java
  bdo-api
  bdo-impl
```

Coding Practices: Integrated Development Environment (IDE)

- Varying features, but often include:
  - Code editor with smart indenting
  - Syntax highlighting
  - Code completion
  - Code inspection
  - Debugger
  - Code refactoring
  - Documentation display
- Linux IDEs include:
  - Eclipse (Java, C++, Python, Fortran...)
  - Netbeans (Java, C++, PHP)
  - MonoDevelop (C#, F#, C/C++...)
  - Emacs can be configured as an IDE (many languages)



Java - bdo/bdo-impl/src/main/java/edu/cornell/soc/bdo/impl/MessagePassingSchedulerImpl.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer

```
bdo
  bdo-api
  bdo-impl
  bdo-model
  bdo-persistence
  bdo-persistence-impl-memory
  bdo-persistence-impl-sqlserver
    src
      main
        java
          edu
            cornell
              soc
                bdo
        resources
          messagePairs.sql
          messages.sql
          nodes.sql
          participantBinding.sql
          participants.sql
          taskInputs.sql
          taskOutputs.sql
          tasks.sql
          topologies.sql
          topologyEdges.sql
    target
    pom.xml
  pom.xml
  bdo-webapp
  pom.xml
  README.md
bdo-api
bdo-impl
bdo-model
bdo-persistence
bdo-persistence-impl-memory
bdo-persistence-impl-sqlserver
bdo-webapp
```

experiment_v... | ExperimentDe... | web.xml | SchedulerBas... | bdo/bdo-weba...

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- This web.xml file is not required when using Servlet 3.0 contain
     see implementation details http://jersey.java.net/nonav/documenta
<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java

    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>classpath:applicationContext.xml</param-value>
    </context-param>

       <listener>
        <listener-class>
            org.springframework.web.context.ContextLoaderListener
        </listener-class>
    </listener>

    <servlet>
        <servlet-name>Jersey Web Application</servlet-name>
        <servlet-class>org.glassfish.jersey.servlet.ServletContainer<
        <init-param>
            <param-name>javax.ws.rs.Application</param-name>
            <param-value>edu.cornell.soc.bdo.web.Application</param-v
        </init-param>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>Jersey Web Application</servlet-name>
        <url-pattern>svc/*</url-pattern>
    </servlet-mapping>
</web-app>
```

Design Source

MessagePassingSchedulerImpl.java

```
final Random prng = new Random();

Set<Integer> nodesInRound = new HashSet<>();

public void init(ExperimentDefinition def,
                 ExperimentRunInfo runInfo,
                 Map<String, String> params) {
    this.experiment = def;
    this.runInfo = runInfo;

    if (def.getSeed() != null) {
        prng.setSeed(def.getSeed());
    }

    loadTaskDefinitionsInOrder(def, taskDefinitions);
}

public List<Node> generateNodesForRound(int round) {
    nodesInNextRound = new LinkedList<>();

    /*
     * Calculate the nodes in the next round, by combining nodes that
     * are children of the previous round (if any) with nodes that ha
     * specified input (and therefore may be newly introduced in this
     * round)
     */
    nodesInNextRound.addAll(findNodesWithSpecifiedInput(experiment,
                                                        round));
    nodesInNextRound.addAll(findChildrenOfNodes(experiment
            .getTopol   d(), nodesInRound));

    /* Now set the cu     node list to what we just calculated */
    nodesInRound.clea
    nodesInRound.addA    nodesInNextRound);
```

Task List

Find | All | Activate...

Connect Mylyn
Connect to your task and ALM tools or create a local task.

Outline

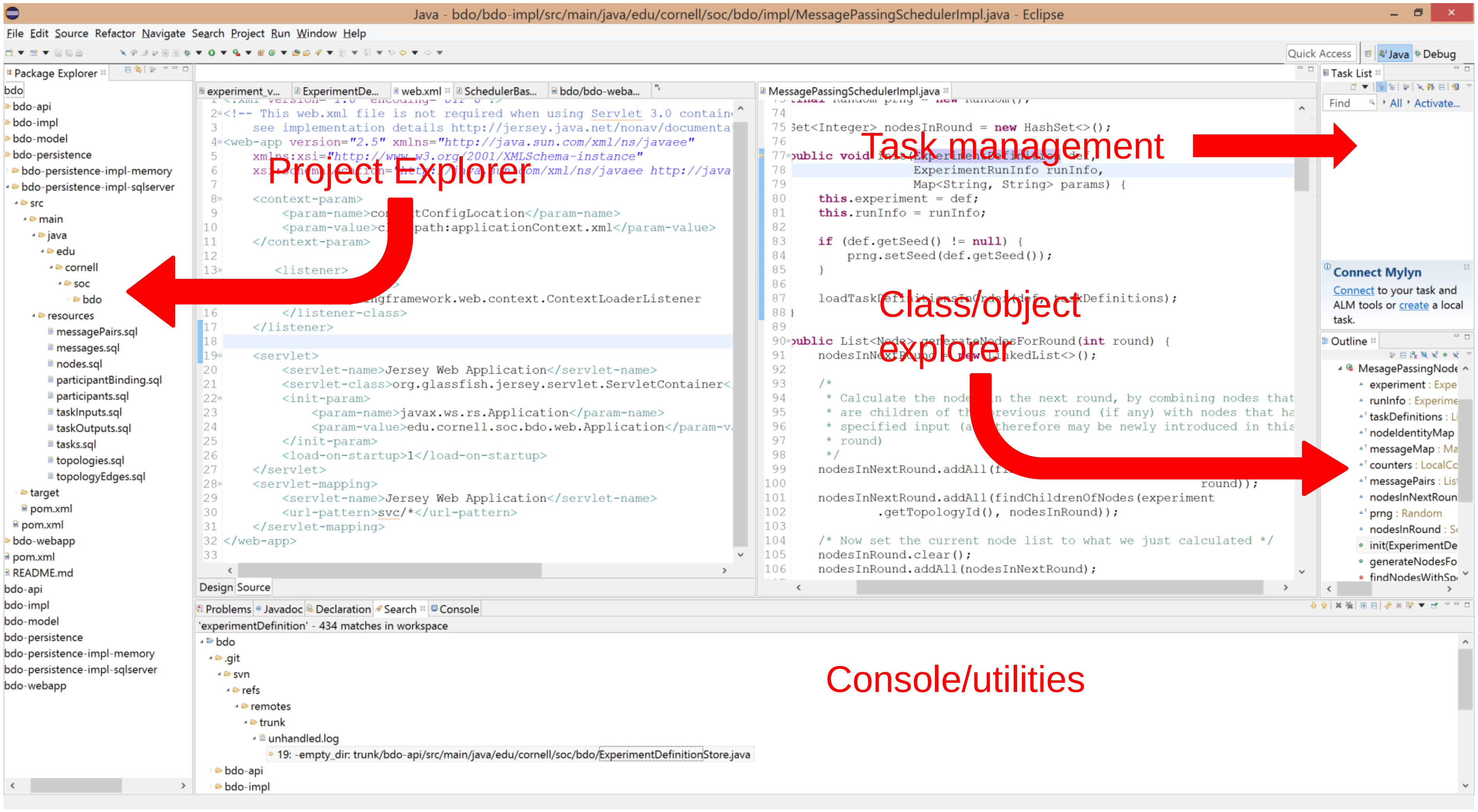
```
MesagePassingNode
  experiment : Expe
  runInfo : Experime
  taskDefinitions : Li
  nodeIdentityMap
  messageMap : Ma
  counters : LocalCo
  messagePairs : Lis
  nodesInNextRoun
  prng : Random
  nodesInRound : Se
  init(ExperimentDe
  generateNodesFo
  findNodesWithSp
```

Problems | Javadoc | Declaration | Search | Cons

'experimentDefinition' - 434 matches in workspace

```
bdo
  .git
    svn
      refs
        remotes
          trunk
            unhandled.log
              19: -empty_dir: trunk/bdo-api/src/main/java/edu/cornell/soc/bdo/ExperimentDefinitionStore.java
  bdo-api
  bdo-impl
```

Editor windows



Java - bdo/bdo-impl/src/main/java/edu/cornell/soc/bdo/impl/MessagePassingSchedulerImpl.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer

```
bdo
  bdo-api
  bdo-impl
  bdo-model
  bdo-persistence
  bdo-persistence-impl-memory
  bdo-persistence-impl-sqlserver
    src
      main
        java
          edu
            cornell
              soc
                bdo
        resources
          messagePairs.sql
          messages.sql
          nodes.sql
          participantBinding.sql
          participants.sql
          taskInputs.sql
          taskOutputs.sql
          tasks.sql
          topologies.sql
          topologyEdges.sql
    target
    pom.xml
  pom.xml
  bdo-webapp
  pom.xml
  README.md
bdo-api
bdo-impl
bdo-model
bdo-persistence
bdo-persistence-impl-memory
bdo-persistence-impl-sqlserver
bdo-webapp
```

experiment_v... | ExperimentDe... | web.xml | SchedulerBas... | bdo/bdo-weba...

```
<!-- This web.xml file is not required when using Servlet 3.0 contain
    see implementation details http://jersey.java.net/nonav/documenta
<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java
    <context-param>
        <param-name>co    tConfigLocation</param-name>
        <param-value>c    path:applicationContext.xml</param-value>
    </context-param>

      <listener>
                     ngframework.web.context.ContextLoaderListener
        </listener-class>
    </listener>

    <servlet>
        <servlet-name>Jersey Web Application</servlet-name>
        <servlet-class>org.glassfish.jersey.servlet.ServletContainer<
        <init-param>
            <param-name>javax.ws.rs.Application</param-name>
            <param-value>edu.cornell.soc.bdo.web.Application</param-v
        </init-param>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>Jersey Web Application</servlet-name>
        <url-pattern>svc/*</url-pattern>
    </servlet-mapping>
</web-app>
```

Design Source

MessagePassingSchedulerImpl.java

```
Set<Integer> nodesInRound = new HashSet<>();

public void init(ExperimentDefinition def,
                 ExperimentRunInfo runInfo,
                 Map<String, String> params) {
    this.experiment = def;
    this.runInfo = runInfo;

    if (def.getSeed() != null) {
        prng.setSeed(def.getSeed());
    }

    loadTaskDefinitionsInOrder(def, taskDefinitions);
}

public List<Node> generateNodesForRound(int round) {
    nodesInNextRound = new LinkedList<>();

    /*
     * Calculate the nodes in the next round, by combining nodes that
     * are children of the previous round (if any) with nodes that ha
     * specified input (and therefore may be newly introduced in this
     * round)
     */
    nodesInNextRound.addAll(fi
                                        round));

    nodesInNextRound.addAll(findChildrenOfNodes(experiment
            .getTopologyId(), nodesInRound));

    /* Now set the current node list to what we just calculated */
    nodesInRound.clear();
    nodesInRound.addAll(nodesInNextRound);
```

Task List

Connect Mylyn

Connect to your task and ALM tools or create a local task.

Outline

```
MesagePassingNode
  experiment : Expe
  runInfo : Experime
  taskDefinitions : Li
  nodeIdentityMap
  messageMap : Ma
  counters : LocalCo
  messagePairs : Lis
  nodesInNextRoun
  prng : Random
  nodesInRound : S
  init(ExperimentDe
  generateNodesFo
  findNodesWithSp
```

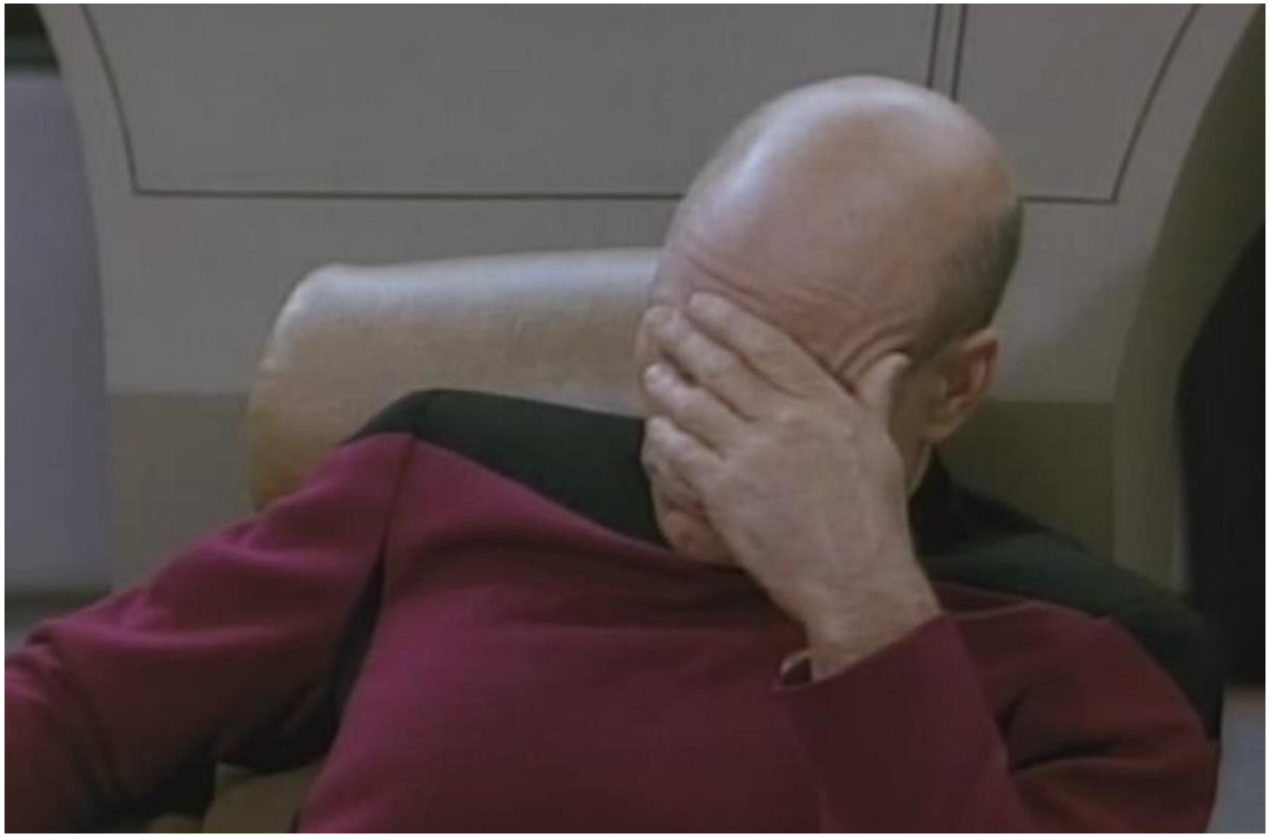
Problems | Javadoc | Declaration | Search | Console

'experimentDefinition' - 434 matches in workspace

```
bdo
  .git
    svn
      refs
        remotes
          trunk
            unhandled.log
              19: -empty_dir: trunk/bdo-api/src/main/java/edu/cornell/soc/bdo/ExperimentDefinitionStore.java
  bdo-api
  bdo-impl
```

Coding Practices: The last change broke the code! Where did I save the previous, working version?

Coding Practices: the last change broke the code! Where did I save the previous, working version?



(Actually, Picard would use versioning)

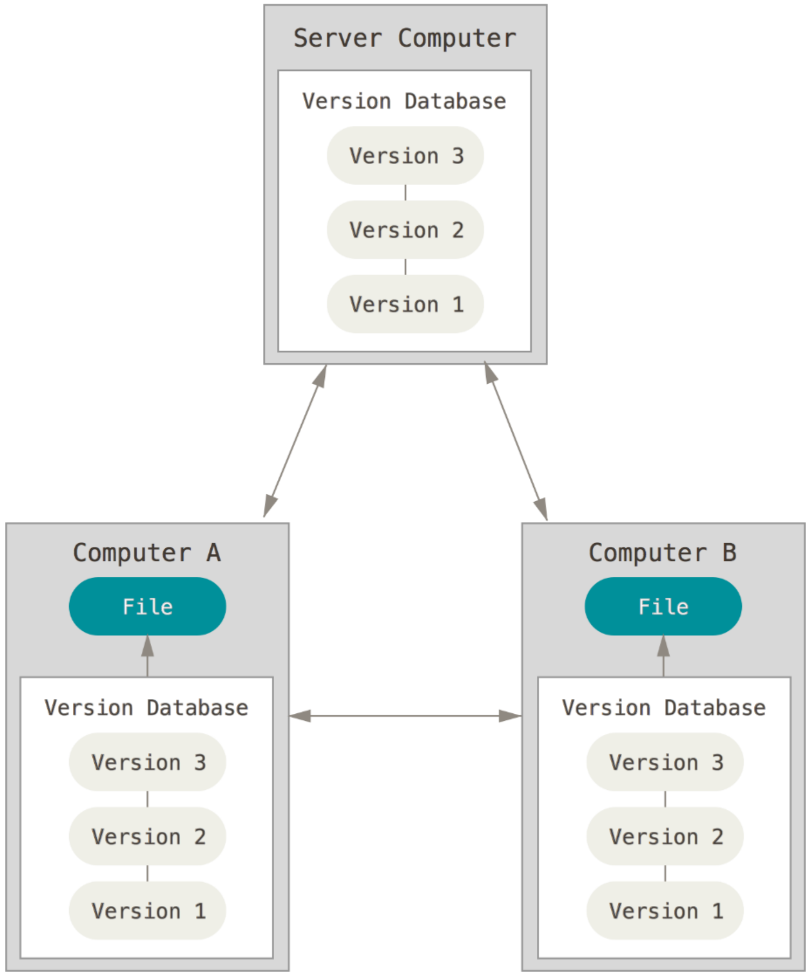
Coding Practices: versioning and source control

- A versioning system typically called a (code) repository

- Versioning has many benefits:
  - Checkpointing development
  - Can roll back to previous versions
  - Can start a branch for development
  - Enables collaborative software development
    - code conflict resolution, distribution to multiple recipients

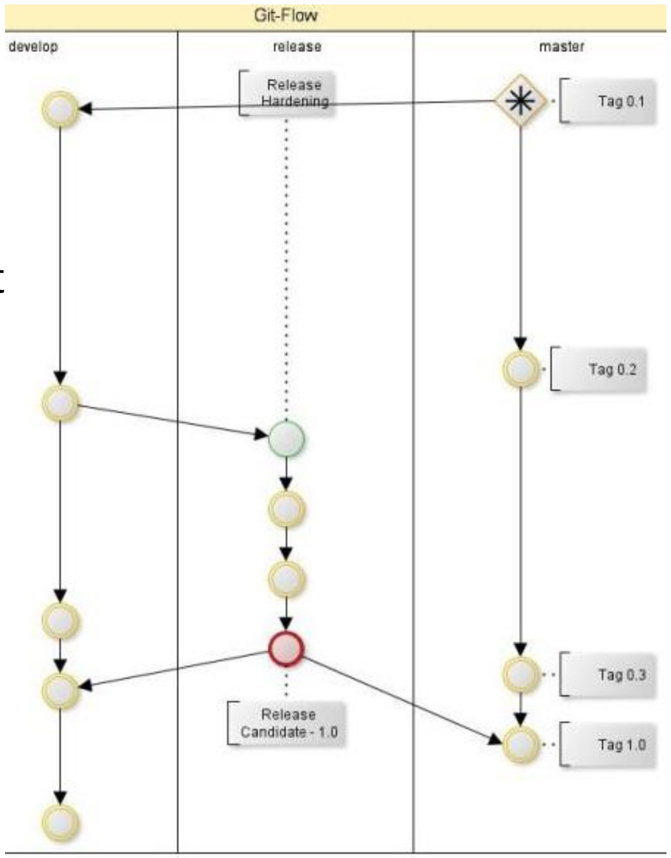
- Most current versioning systems:
  - Show diffs between committed code versions, highlight code conflicts, enable merges
  - Have command-line and graphical interfaces, IDE integration

Coding Practices: what is distributed versioning?



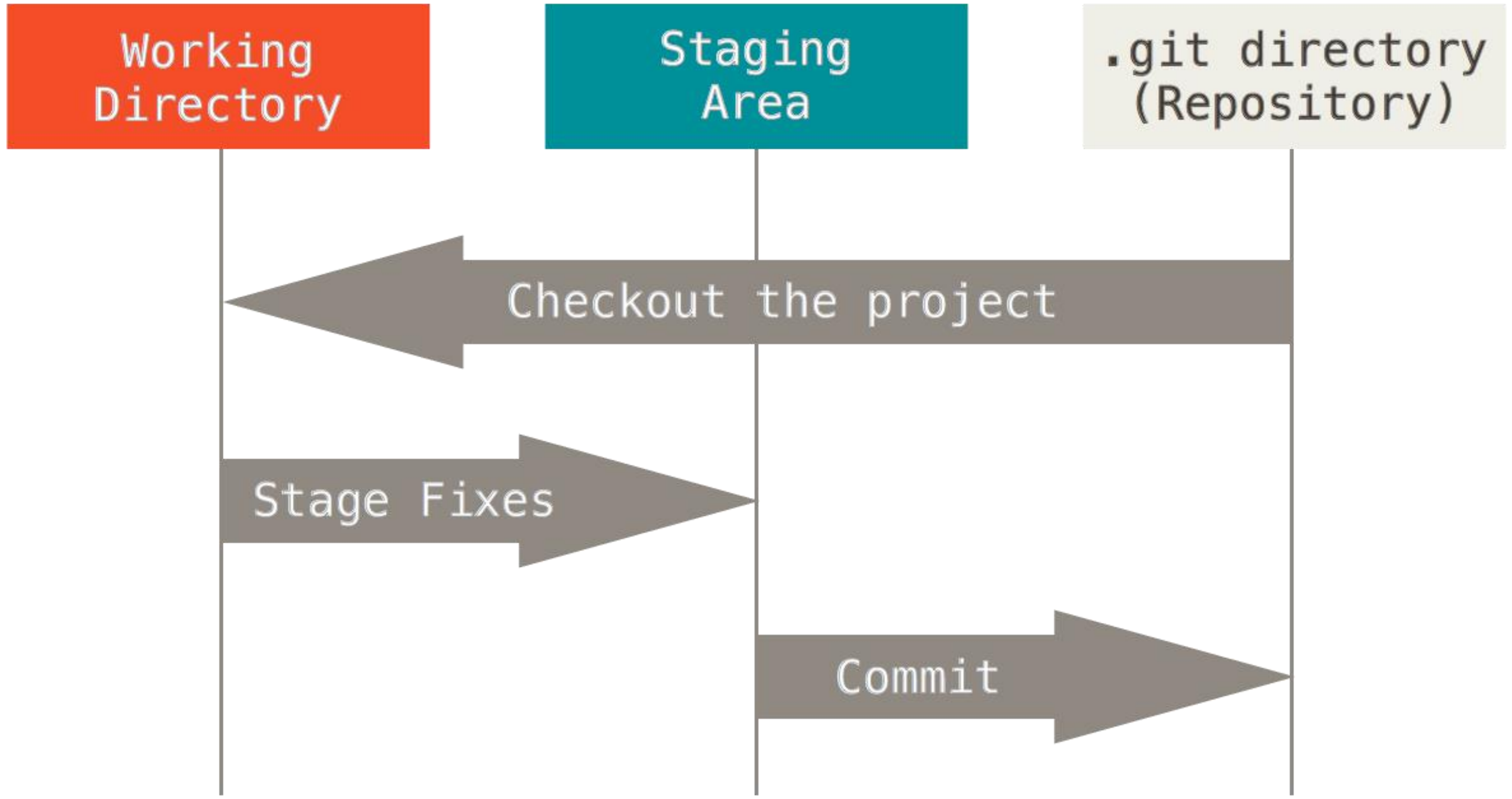
Coding Practices: versioning decisions, decisions.

- Common versioning systems:
  - Older:
    - CVS: now somewhat venerable but still in use
    - Subversion (SVN): common CVS replacement
  - Distributed version control
    - Mercurial: focus on usability
    - Git: powerful, developed by linux kernel team
  - Hosting with web-based interfaces include:
    - Sourceforge (SVN, Git, Mercurial)
    - Bitbucket (Git, Mercurial)
    - Github (Git)
- Git installed on Stampede:
  - Load the Git module to get a recent version



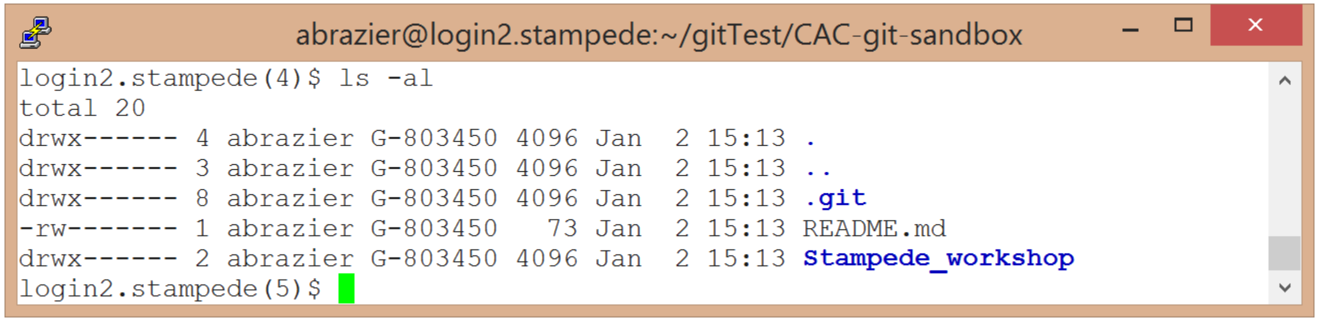
Branches, commits and tags

Coding Practices: Git exercise. What is Git doing?



Coding Practices: Git exercise. Create repository

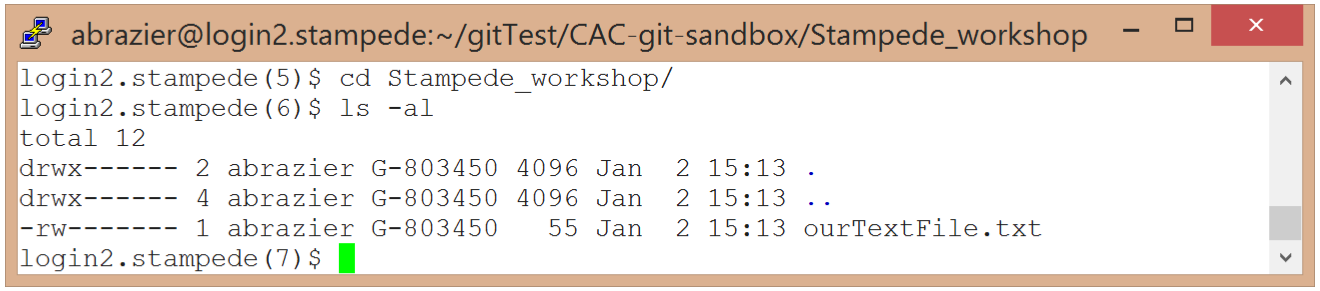
- Log into Stampede.
  - `mkdir gitTest`
  - `load module git`
  - `cd gitTest`
  - `git clone https://github.com/cornell-comp-internal/CAC-git-sandbox.git`
  - `cd CAC-git-sandbox/`
  - `ls -a`


abrazier@login2.stampede:~/gitTest/CAC-git-sandbox

```
login2.stampede(4)$ ls -al
total 20
drwx------ 4 abrazier G-803450 4096 Jan  2 15:13 .
drwx------ 3 abrazier G-803450 4096 Jan  2 15:13 ..
drwx------ 8 abrazier G-803450 4096 Jan  2 15:13 .git
-rw------- 1 abrazier G-803450   73 Jan  2 15:13 README.md
drwx------ 2 abrazier G-803450 4096 Jan  2 15:13 Stampede_workshop
login2.stampede(5)$
```

Coding Practices: Git exercise. Setup, editing a document

- `git config --global user.name "<your name>"`
- `git config --global user.email "<your email>"`
- `cd Stampede_workshop`
- `ls`



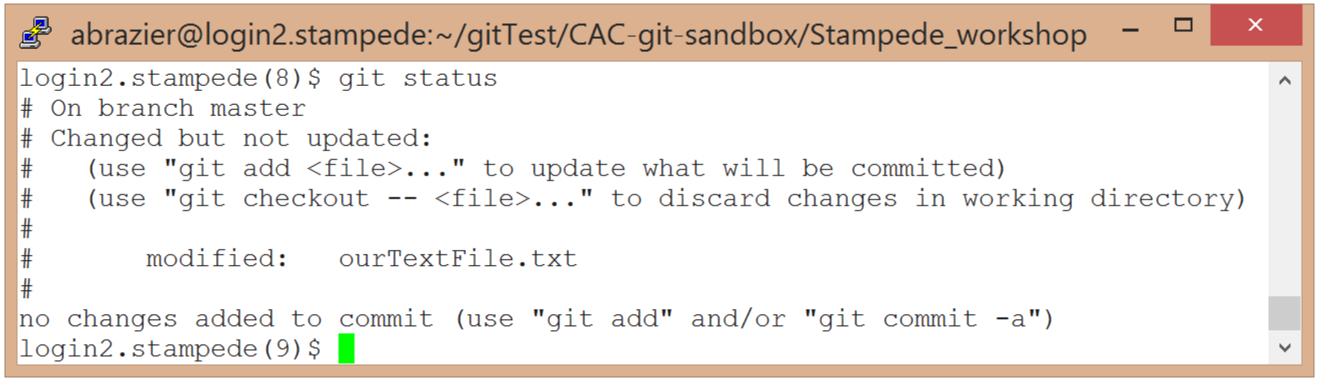
abrazier@login2.stampede:~/gitTest/CAC-git-sandbox/Stampede_workshop

```
login2.stampede(5)$ cd Stampede_workshop/
login2.stampede(6)$ ls -al
total 12
drwx------ 2 abrazier G-803450 4096 Jan  2 15:13 .
drwx------ 4 abrazier G-803450 4096 Jan  2 15:13 ..
-rw------- 1 abrazier G-803450   55 Jan  2 15:13 ourTextFile.txt
login2.stampede(7)$
```

- Open the file `ourTextFile.txt`
  - Edit in a line at the end (end with newline)
  - Close and save

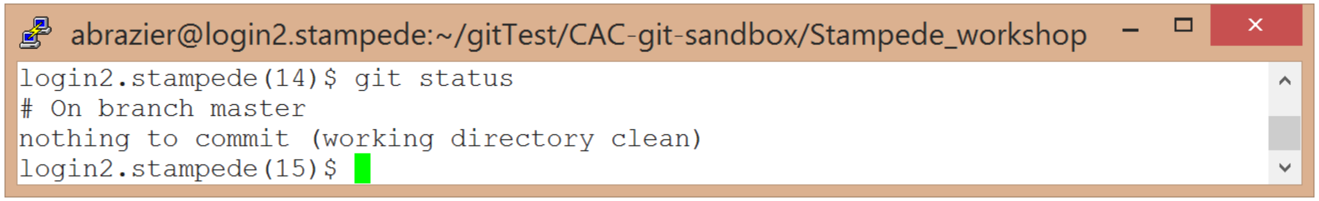
Coding Practices: Git exercise. Status, committing

- `git status`



```
abrazier@login2.stampede:~/gitTest/CAC-git-sandbox/Stampede_workshop
login2.stampede(8)$ git status
# On branch master
# Changed but not updated:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       modified:   ourTextFile.txt
#
no changes added to commit (use "git add" and/or "git commit -a")
login2.stampede(9)$
```

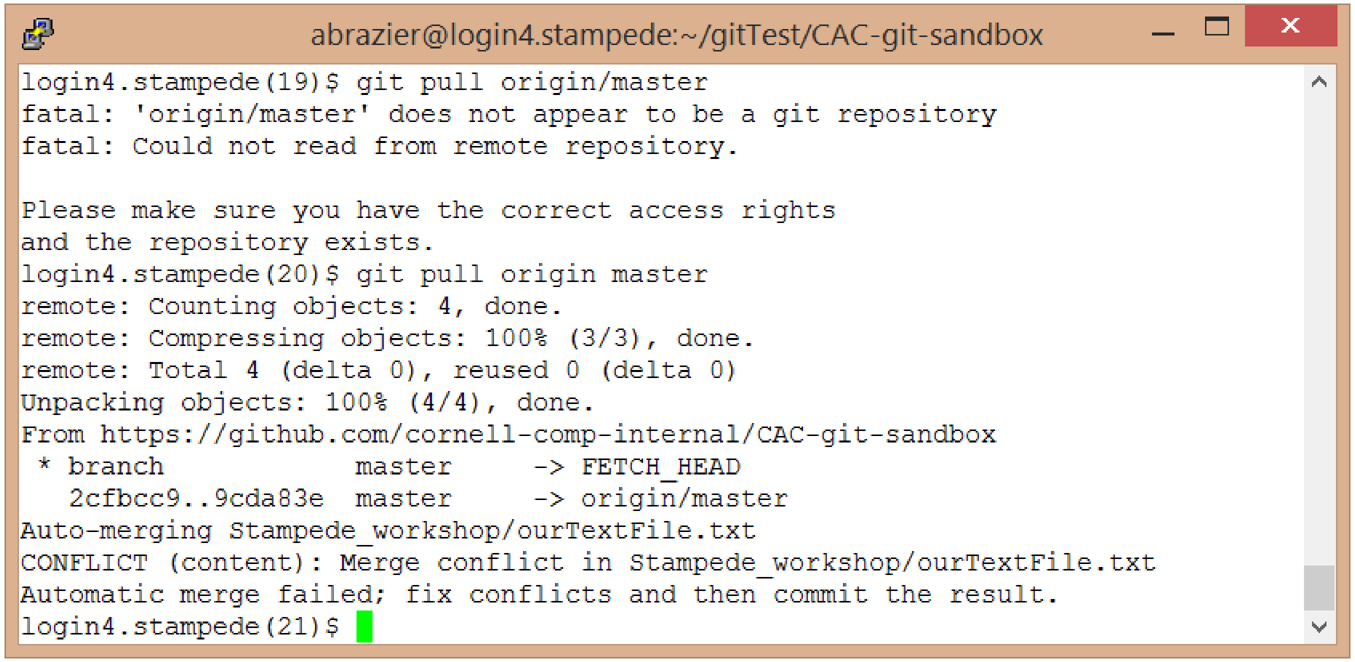
- `git add ourTextFile.txt`
- `git commit –m ”<your commit message>”`
- `git status`



```
abrazier@login2.stampede:~/gitTest/CAC-git-sandbox/Stampede_workshop
login2.stampede(14)$ git status
# On branch master
nothing to commit (working directory clean)
login2.stampede(15)$
```

Coding Practices: Git exercise. Oh no, a conflict!

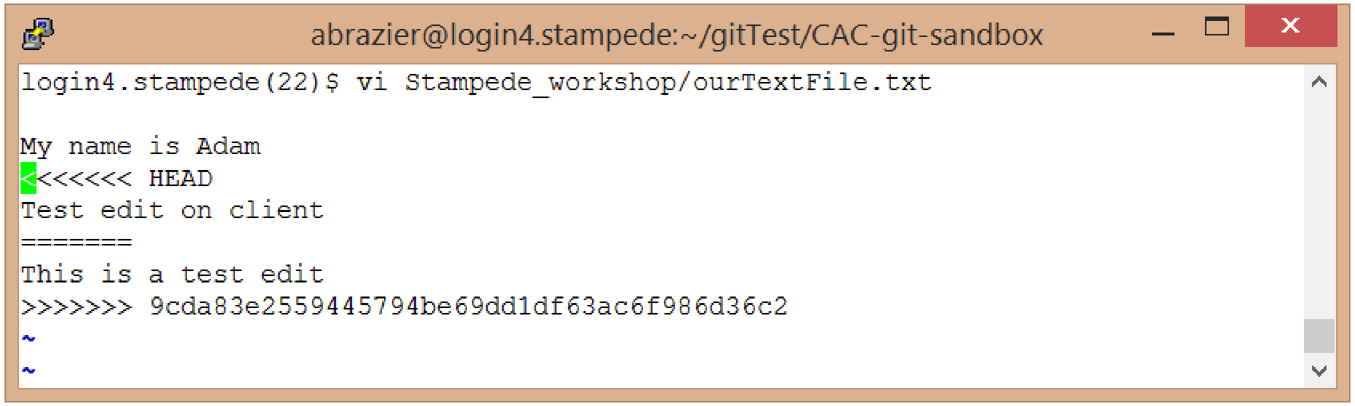
- I will make some changes to the `ourTextFile.txt` on Github
- We now have a conflict between the two versions
- `git pull origin master`



```
login4.stampede(19)$ git pull origin/master
fatal: 'origin/master' does not appear to be a git repository
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.
login4.stampede(20)$ git pull origin master
remote: Counting objects: 4, done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 4 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (4/4), done.
From https://github.com/cornell-comp-internal/CAC-git-sandbox
 * branch            master     -> FETCH_HEAD
   2cfbcc9..9cda83e  master     -> origin/master
Auto-merging Stampede_workshop/ourTextFile.txt
CONFLICT (content): Merge conflict in Stampede_workshop/ourTextFile.txt
Automatic merge failed; fix conflicts and then commit the result.
login4.stampede(21)$
```

Coding Practices: Git exercise. Resolving conflicts

- Open `ourTestFile.txt` for editing

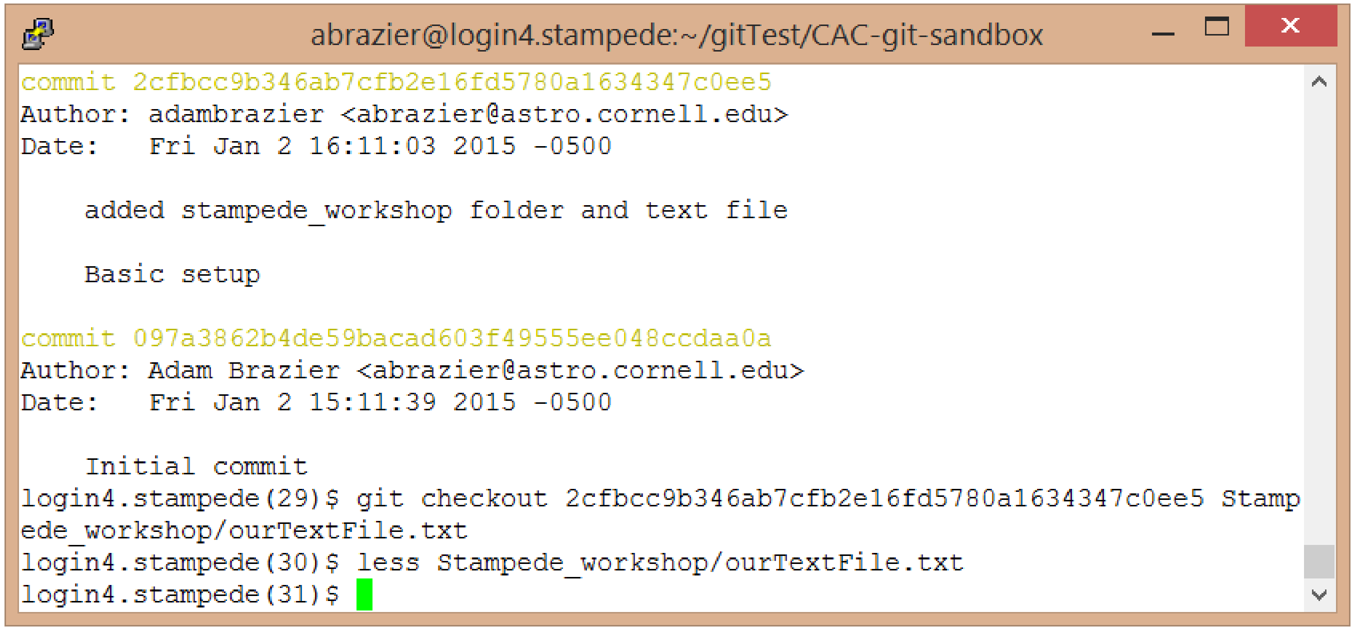


```
login4.stampede(22)$ vi Stampede_workshop/ourTextFile.txt

My name is Adam
<<<<<<< HEAD
Test edit on client
=======
This is a test edit
>>>>>>> 9cda83e2559445794be69dd1df63ac6f986d36c2
~
~
```

- Resolve the conflict between <<<< and >>>> SHA1
  - Be sure to remove those as well
- `git add Stampede_workshop/ourTestFile.txt`
- `git commit -m "<your commit message>"`

Coding Practices: Git exercise. I preferred the old version!

- Use `git log` to find the commit SHA1 from the initial commit
- `git checkout <SHA1> Stampede_workshop/ourTestFile.txt`
- Read the file! You can `git commit` to make the reversion stick



```
commit 2cfbcc9b346ab7cfb2e16fd5780a1634347c0ee5
Author: adambrazier <abrazier@astro.cornell.edu>
Date:   Fri Jan 2 16:11:03 2015 -0500

    added stampede_workshop folder and text file

    Basic setup

commit 097a3862b4de59bacad603f49555ee048ccdaa0a
Author: Adam Brazier <abrazier@astro.cornell.edu>
Date:   Fri Jan 2 15:11:39 2015 -0500

    Initial commit
login4.stampede(29)$ git checkout 2cfbcc9b346ab7cfb2e16fd5780a1634347c0ee5 Stamp
ede_workshop/ourTextFile.txt
login4.stampede(30)$ less Stampede_workshop/ourTextFile.txt
login4.stampede(31)$ 
```

Coding Practices: Distributed version control with Git

- Pushing back to a remote repository (including Github) is generally done with `git push`
- Unless someone is waiting on your version, it's normally OK to push at end of day, but every team has different policies
- Local commits should be done relatively frequently
- Git is not an alternative to backups, but should be integrated into a backup solution (Github can be part of that)

Coding Practices: learning Git

- www.git-scm.com (scm = “source code management”)
  - Contains the text of the book “Pro Git” at www.git-scm.com/book

- Stack Overflow, etc (useful if you have a specific error message)

- https://github.com will typically give you five free private repositories if you work in academic research (and have a .edu email address)
  - Go to https://education.github.com and make a request
  - Github gives each repository a markdown wiki
  - Github has functionality for tracking issues
  - Github’s pages also contain some Git advice
  - Github has specific GUI clients which also take care of authentication

Overview

- Coding practices
  - (because most of us are already coding)
- Software Lifecycle
  - Cradle to Grave
- Improving the Software
  - Faster. Better. Stronger

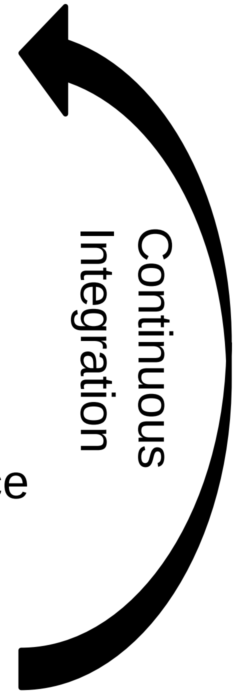
Software Lifecycle

- The ages of software
  - Requirements
  - Design
  - Implementation
  - Testing/QA/acceptance
  - Documentation

Cradle to grave

Software Lifecycle

- The ages of software
  - Requirements
  - Design
  - Implementation
  - Testing/QA/acceptance
  - Documentation



Cradle to grave

Software Lifecycle: Requirements. I want it all!

- Requirements are like the contract between user and coding team
  - In the research environment both groups may be the same people; arguably requirements are as important in this case as any other

- Requirements *must* be testable
  - A good check is to ask “how can I check if this requirement is met?”
    - If you can’t answer that question, it’s not a requirement.

- Requirements may be split into two groups:
  - Functional requirements
  - Design constraints/non-functional requirements

Software Lifecycle: anatomy of a requirement

- Requirements are typically of this form:

  The X shall Y

- For example:
  - "The calendar widget shall allow selection of discontiguous dates"
    - (this is an example of a functional requirement)
  - "The software system will average at least 1.1 GB of data processing per minute, 95% of the time"
    - (this is an example of a design constraint/non-functional requirement)

- Both of these requirements are *testable.*

Software Lifecycle: pitfalls and perils

- Every requirement, in effect, costs time/money
  - Not a negative statement; software costs time and money to make!
  - The set of requirements should be necessary (obviously!)
    - And sufficient…
- Changing the requirements can cost a lot of additional time/money
  - Can cause substantial rework:
    - Re-design and re-architecting
    - Re-coding
  - Sometimes requirement changes are needed
- Design, Architecture, implementation and test plans should be linked to requirements

Software Lifecycle: some questions

- All software effort exists in a context
  - Who will use this software?
  - What resources are available or can be sought?
  - How long might the product last?
  - Will it support publications?
  - What are our priorities?
- Requirements and design must be consistent with the answers to these questions.
  - Perhaps the worst outcome to a completed software development project is to have produced an unusable product or a “solution looking for a problem”.

Software Lifecycle: software design

- Once you know your software requirements, you must design to meet them

- Design elements include (loose order):
  - Framework (including languages, testing/QA, Continuous Integration)
  - Architecture
  - Software components (including re-use)
  - Communication
  - Data structures
  - Algorithms

But is it comfortable?

Software Lifecycle: software design

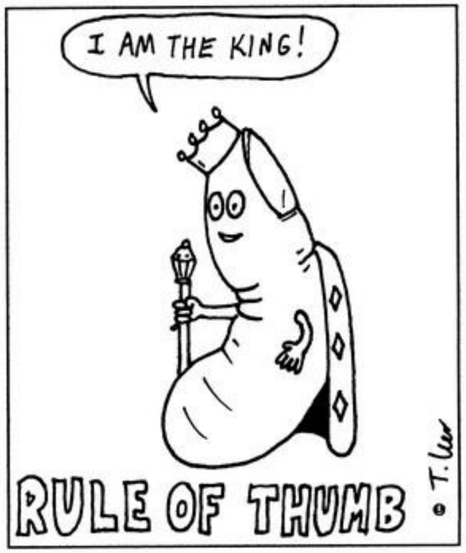
- Once you know your software requirements, you must design to meet them
- Design elements include (loose order):
  - Framework (including languages, testing/QA, Continuous Integration)
  - Architecture
  - Software components (including re-use)
  - Communication
  - Data structures
  - Algorithms

But is it comfortable?

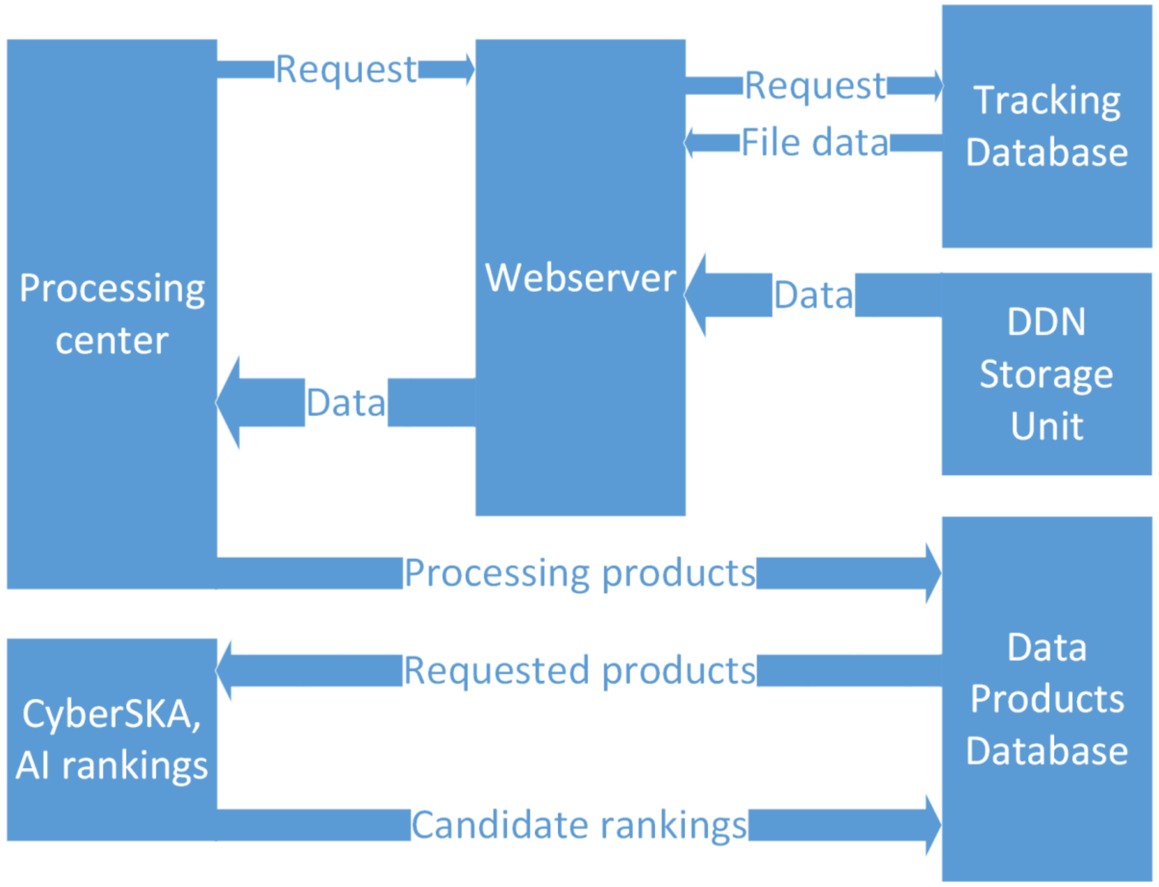
- “Days of programming can save hours of thinking!”

Software Lifecycle: some guidelines

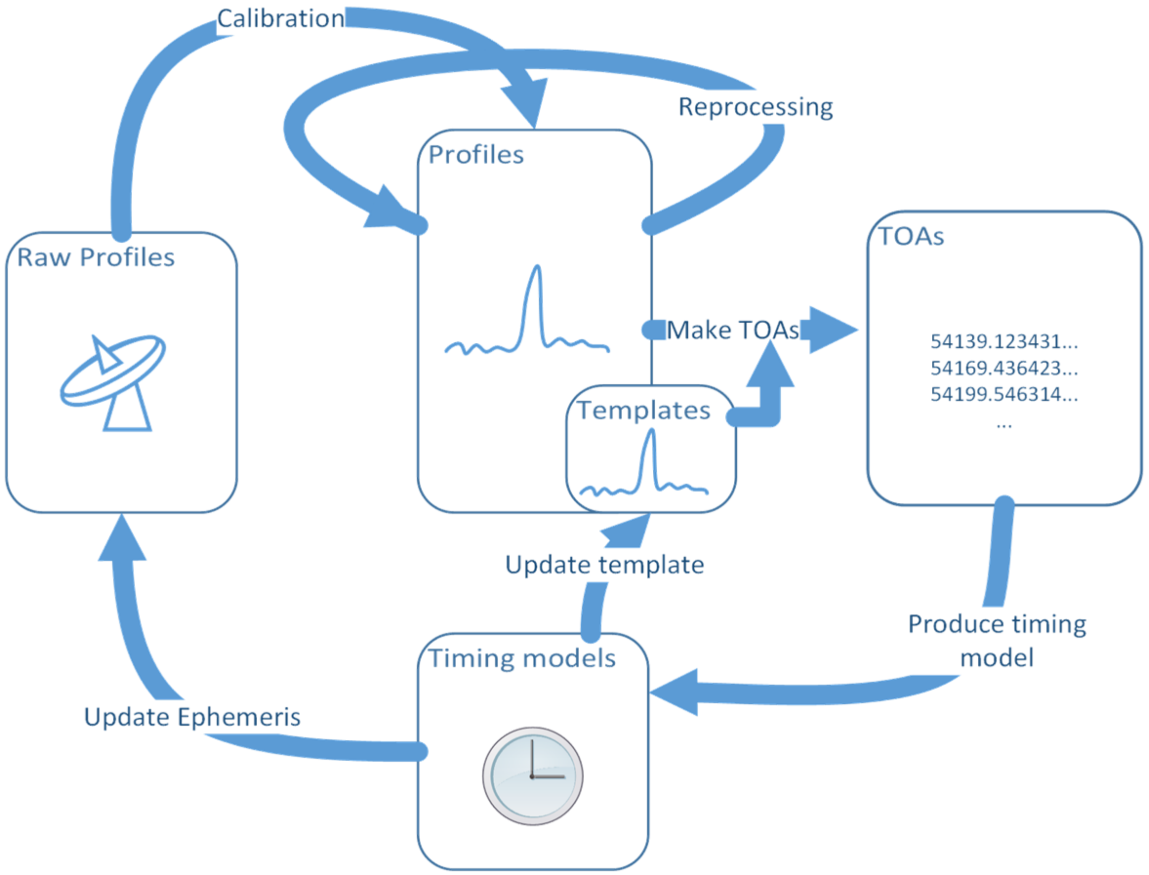
- Your design, if implemented, is a template for the documentation
- Diagrams are very helpful
- Modularity and layering allow abstraction
  - simplifies implementation and maintenance
- Design Review is important
- Identify how it satisfies the requirements
  - Particularly helpful if requirements change



Software Lifecyle: diagrams visualize flow

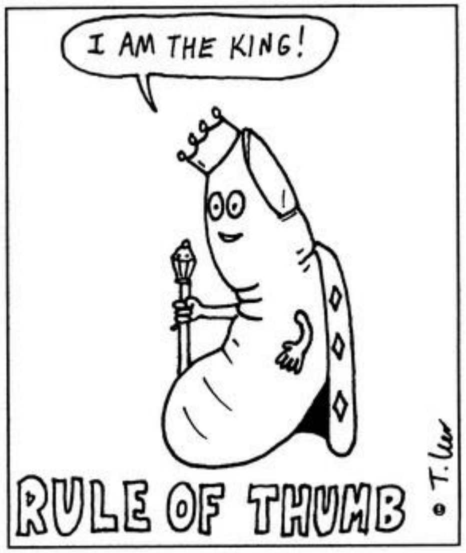


Software Lifecycle: diagrams show functionality



Software Lifecycle: some guidelines

- Your design, if implemented, is a template for the documentation
- Diagrams are very helpful
- Modularity and layering allow abstraction
  - simplifies implementation and maintenance
- Design Review is important
- Identify how it satisfies the requirements
  - Particularly helpful if requirements change



Software Lifecycle: implementation

- Common implementation methodologies today stress:
  - Use cases/user stories
  - Fast development cycles
  - Whole-team awareness
  - Prototyping
  - Regular meetings
  - Mixed expertise
  - Integrated testing

Agile!

Software Lifecycle: implementation

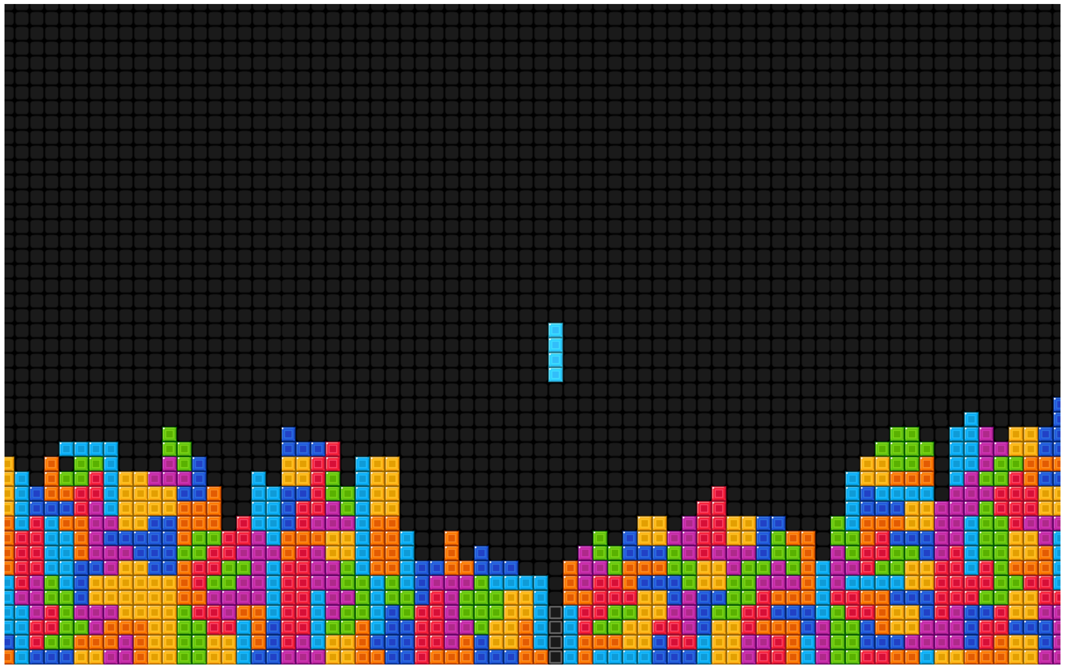
- Common implementation methodologies today stress:
  - Use cases/user stories
  - Fast development cycles
  - Whole-team awareness
  - Prototyping
  - Regular meetings
  - Mixed expertise
  - Integrated testing

- Sounds a lot like the research environment!

Agile!

Software Lifecycle: coding

- Already covered coding practices but:



- All this stuff fits together!

Software Lifecycle: software estimation

- *"The first 90 percent of the code accounts for the first 90 percent of the development time. The remaining 10 percent of the code accounts for the other 90 percent of the development time."*
  - *Tom Cargill's ninety-ninety rule*

- One simple approach:
  - Identify all project elements, including testing and documentation!
  - Estimate each individually, including a margin of error
  - Sum the estimates

- Keep alert to changes in requirements and to progress in timeline.
  - Identifying problems early gives more freedom to plan remediation.

Software Lifecycle: deployment and upgrades

- Software often deployed on machines on which it wasn’t developed

- Developers plan deployment. Typically, on Linux:
  - README: gives version information and help for installation
  - Configuration Script (typically called `configure`), optional
    - May be written by hand, generated by `autoconf`, etc
  - Build instructions
    - Often done with `Make`
    - Languages may have their own tools, eg, maven, ez_install
    - Environment virtualization can make this easier

Software Lifecycle: does it work?

- In the research environment, most important elements include:
  - Levels
    - Unit Tests. Typically written as code is developed, can be executed at build time
    - Interface and interoperability
    - System testing: covers the entire product, top to bottom. Tests driven by the requirements
  - Types
    - Build testing: will it run where it's being built?
    - Functional tests: tests required functional behavior of components
    - Performance testing: does it meet performance requirements? How does performance scale with load?
    - Regression testing: testing that changes/upgrades to the software haven't broken anything
    - Operational testing: does it work when you run it a lot, with different inputs?

Overview

- Coding practices
  - (because most of us are already coding)
- Software Lifecycle
  - Cradle to Grave
- Improving the Software
  - Faster. Better. Stronger

Improving the Code: making applications better

- Upgrading means integration of new code into existing software

- Modern software development typically does this frequently
  - Continuous Integration (CI) involves very frequent integrations into the main build and, typically, frequent releases
  - Automated testing a key element of CI on typical scales

- In HPC, putting out new versions without testing first can waste a lot of time/money very quickly
  - Code should be tested before deployment. This may be a fair amount of development just towards testing.

It’s never done!

- There are several reasons to change a software product, including:
  - Adding features
  - Correcting bugs
  - Improving performance
  - Changing external interfaces
  - Changing environment
    - Includes infrastructure upgrades like advancing language version
- Managing this process can be similar to Agile development
- Suggestion: use semantic versioning to label releases
  - www.semver.org

Benchmarking and Optimization

- Benchmarking
  - Forms baseline for performance evaluation
  - Used to test performance requirements are met
  - Requires documenting the hardware and context
  - Can be re-run as supplement to regression testing
- Profiling
  - Key component of understanding code performance
  - Part of the benchmark
- Optimization will be dealt with in its own talk

This one goes to 11

Summary

- Incorporation of new practices should be needs-based
- You don’t have to do it all at once!
- Think of code as a document, read by humans as well as compilers
- Best practices tend to be intercompatible and mutually supporting
- Try some things out!