Activity #25: Version Control

Recorder's Report

Manager:		Reader:			
Recorder:		Driver:			
Date:		Score:	Satisfactory	/	Not Satisfactory

Record your team's answers to the key questions (marked with 🔑) below.

a) Model 1, Question #4

b) Model 2, Question #8

Activity #25: Version Control

In this activity, you will work in teams of 3–4 students to learn new concepts. This activity will introduce you to version control.

Content Learning Objectives

After completing this activity, students should be able to:

- Explain the core VCS concepts of *checking out*, *updating*, *branching*, and *merging*
- Describe an effective workflow for a VCS

Process Skill Goals

During the activity, students should make progress toward:

- Construct timeline diagrams depicting various interactions with a VCS

Model 1 Version Control Scenarios

$$\begin{tikzcd}
 & A_n \\
V_n \arrow[ur, dashed] \arrow[dr, dashed] & \\
 & B_n
\end{tikzcd}$$

Scenario #1

$$\begin{tikzcd}
 & A_n \arrow[dr, dashed] & \\
V_n \arrow[ur, dashed] \arrow[rr] \arrow[dr, dashed] & & V_{n+1} \\
 & B_n &
\end{tikzcd}$$

Scenario #2

$$\begin{tikzcd}
 & A_n \arrow[dr, dashed] & & & \\
V_n \arrow[ur, dashed] \arrow[rr] \arrow[dr, dashed] & & V_{n+1} \arrow[rr] \arrow[dr, dashed] & & V_{n+2} \\
 & B_n \arrow[rr, dashed] & & B_{n+1} \arrow[ur, dashed] &
\end{tikzcd}$$

Scenario #3

Refer to Model 1 above as your team develops consensus answers to the questions below.

Questions (25 min) Start time:

1. A *version control system* (VCS) helps to manage multiple versions of files, such as program source code. Such a collection of files is called a *repository*. In the model above, V_n represents the current version of the repository. If Amy and Bob are two developers working on the code stored in V_n, they will first *check out* their own local copies, A_n and B_n, of the repository to edit. The first scenario of the model depicts this situation.

a) What do the dashed arrows represent?

b) Why is it better for Amy and Bob to edit their own copies of the repository rather than files in V_n directly?

c) What problems might result from Amy and Bob editing their own copies of the repository?

2. After editing, Amy adds her changes to the main repository to create a new version, V_{n+1}. We say that Amy *commits* changes to the repository. This is shown in the second scenario.

a) How is Amy's commit shown in the diagram for scenario #2?

b) Before Amy commits her code, can Bob view Amy's changes? Explain.

c) Why is it important that Amy test her code before she commits it?

3. Before Bob can commit his changes, he must get all of Amy's changes from V_{n+1}. We say that Bob *updates* his local copy from the repository. This is depicted in the third scenario.

a) Why must Bob update his local copy before he can commit changes to the repository?

b) Why are there two arrows pointing to B_{n+1}, Bob's updated copy of the repository, instead of one?

c) What should Bob do before he commits his updated changes back to the main repository?

4. Which of the operations discussed in this model (*checkout*, *commit*, and *update*) are best described by the following?

a) Transfers the most amount of data:

b) Transfers the least amount of data:

c) Might result in a *conflict* between file versions:

5. We can use *timeline diagrams* such as those shown in the model to visualize versions (*nodes* represented by rectangles) and connections (*links* represented by arrows). Draw a timeline diagram to show the following sequence of events.

a) We start with version 10 of the project files
b) Amy checks out her own copy
c) Amy edits and commits, creating a new version
d) Bob checks out his own copy
e) Amy makes another edit and commits again
f) Bob updates
g) Bob edits and commits

Model 2 Branching and Merging

$$\begin{tikzcd}
\boxed{V_{1.0}} \arrow[r] \arrow[dr] & \boxed{V_{1.1}} \arrow[r] & \boxed{V_{1.2}} & & \\
 & \boxed{V_{2.A}} \arrow[r] & \boxed{V_{2.B}} \arrow[r] & \boxed{V_{2.0}} \arrow[r] & \boxed{V_{2.1}}
\end{tikzcd}$$

Scenario #1

$$\begin{tikzcd}
\boxed{V_{1.0}} \arrow[r] \arrow[dr] & \boxed{V_{1.1}} \arrow[r] & \boxed{V_{1.2}} \arrow[r] & \boxed{V_{2.0}} \\
 & \boxed{V_{A.1}} \arrow[r] & \boxed{V_{A.2}} \arrow[ur] &
\end{tikzcd}$$

Scenario #2

Refer to Model 2 above as your team develops consensus answers to the questions below.

Questions (25 min) **Start time:**

6. After an initial version of a program is finished, you may wish to start working on new features without breaking the existing version. Versions that have not yet been released are sometimes labeled with the names alpha and beta. You may then continue to make small bug fixes to the existing version in parallel with the major changes being made to the new version. These parallel versions of the software are called *branches* and creating a new parallel version is called *branching*.

a) Identify the initial version of the repository shown in scenario #1 of the model along with the versions that are bug fixes of that initial version.

b) How are versions $V_{1.0}$ and $V_{2.A}$ of this first scenario related?

c) How are versions $V_{1.2}$ and $V_{2.0}$ related?

d) Create a timeline diagram starting with scenario #1 of the model and showing the following additional actions.

(a) A new release is added to the V_1 branch
(b) A new release is added to the V_2 branch
(c) A new branch with versions $3.A$, $3.B$, and 3.0 is added.

7. When multiple people are working on the same repository, it is common practice for a group of them to create a branch in which to add and test their code and then *merge* that branch back into the main branch (called the *trunk*) when they are done. An example of this is shown in scenario #2.

a) Why might a development team choose to add new features in a branch instead of in the trunk?

b) What challenges might the programmers face when merging their branch back into the trunk? How could these challenges be minimized?

c) Draw a timeline diagram starting with scenario #2 of the model that shows another team working on the same project creating branch B based on version 1.1 of the trunk, creating versions B.1 and B.2, and then merging it into version 2.0.

8. Add nodes and/or arrows to the diagram below to create a timeline diagram for each of the following sequence of events.

$$\begin{tikzcd}\boxed{V_{10}} \arrow[r] & \boxed{V_{11}} \arrow[r] & \boxed{V_{12}}\end{tikzcd}$$

a) Version 10 is finished and version 11 is in progress, but defects in version 10 must be fixed and released (as version 10.A) before version 11 is ready.

b) Version 10.A is merged into the main branch before version 11 is released.

c) Version 11 is completed and version 12 is in progress, but some developers are exploring a different approach to error handling (version 11.A), which might be included in version 12, but might also be delayed until version 13.

d) The two developers working on version 11.A disagree on how best to proceed. One finishes a smaller set of changes (as version 11.B) that are added before version 12 is finished. The other continues with a larger set of changes, now called version 11.C, that will be added to version 13.

9. What seems harder, technically – branching or merging? Explain why.

10. One popular modern version control system is called `Git`. Because `Git` is a *distributed* VCS, it it uses slightly different terminology from that developed above. Conduct some internet searches to determine the `git` commands that would accomplish each of the following.

a) Initialize a new repository

b) Create a local copy of a remote repository

c) Check the status of the files in your local repository

d) Add files to your local repository

e) Commit changes to your local repository

f) Update your local repository from a remote repository

g) Send your local changes to the remote repository

h) Create a new branch in your local repository