Option #1: Mutually Conflicting Requirements

Scott Miner

Colorado State University – Global Campus

Abstract

Figure 1. Communication diagram for the conflicting requirements application

```
There was one conflict regarding tab orientation and you WON the coin toss!

There was one conflict regarding the toolbar and you LOST the coin toss!
Updating project specifications.

There was one conflict between four and eight menu items and you WON the coin toss!

**************************************************
*   Creating and opening specifications image...   *
**************************************************

Returning to main menu...
```

Figure 2. Output showing coin flips settling the results of three two-way ties between tab, toolbar, and menu item requirements

```
There was one conflict regarding tab orientation and you WON the coin toss!

There was a three-way tie between the number of menu items.
You roll a three-sided die and return a '3', indicating that there will be eight menu items.
Updating specifications as needed.

**************************************************
*   Creating and opening specifications image...   *
**************************************************
```

Figure 3. Output showing a die roll settling the results of a three-way tie between menu item requirements

```
Welcome to the Software Specifications Application.
Please choose from the choices below.

************************************
*                                  *
*   [1] Register New User          *
*   [2] Log in                     *
*   [0] Exit the program           *
*                                  *
************************************

Enter your option: 1

**********************************
*     Registering a new user     *
**********************************

Enter your email (0 to exit): testemail@

Email must be valid.

Enter your email (0 to exit): testemail@gmail.com

Username is available.

Enter your 4-digit pin number (0 to exit): 323

Pin must contain four digits.

Enter your 4-digit pin number (0 to exit): 1234

**************************
*     New User Added     *
**************************
```

Figure 4. Registering a new user, validating an email, and validating a 4-digit PIN

```
Welcome to the Software Specifications Application.
Please choose from the choices below.

****************************************
*                                      *
*    [1] Register New User             *
*    [2] Log in                        *
*    [0] Exit the program              *
*                                      *
****************************************

Enter your option: 2

************************
*     Log in Screen    *
************************

Enter your email (0 to exit): testemail@gmail.com

Enter your 4-digit pin number (0 to exit): 4444

Pin does not match pin on file.
Please try again.

Enter your 4-digit pin number (0 to exit): 1234

*********************************************
*   Welcome to the Specification Questions   *
*********************************************
```

Figure 5. Logging in once the user enters the correct PIN

```
*********************************************
*   Welcome to the Specification Questions   *
*********************************************

1. How many menu options should the program contain?

*****************************************************************************************
*                                                                                       *
*   [1] 4 (File, Repository, Command, Help)                                             *
*   [2] 6 (File, Repository, Command, Tools, Plugins, Help)                             *
*   [3] 8 (File, Repository, Command, Tools, View, Navigate, Plugins, Help)             *
*   [0] Exit the Program                                                                *
*                                                                                       *
*****************************************************************************************

Enter your option: 3

2. Do you want the program to contain horizontal or vertical tabs?

************************************
*                                  *
*   [1] Vertical                   *
*   [2] Horizontal                 *
*   [0] Exit the program           *
*                                  *
************************************

Enter your option: 2

3. Should the application contain a toolbar?

************************************
*                                  *
*   [1] Yes                        *
*   [2] No                         *
*   [0] Exit the Program           *
*                                  *
************************************

Enter your option: 1

************************************************
*   Creating and opening specifications image...   *
************************************************

Returning to main menu...
```

Figure 6. Prompting for user software specifications

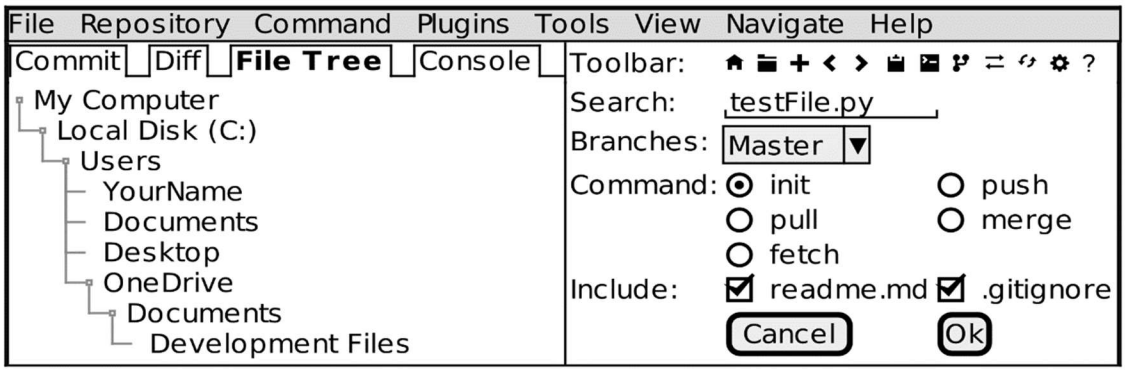

Figure 7. Application output with eight menu options, horizontal tabs, and a toolbar

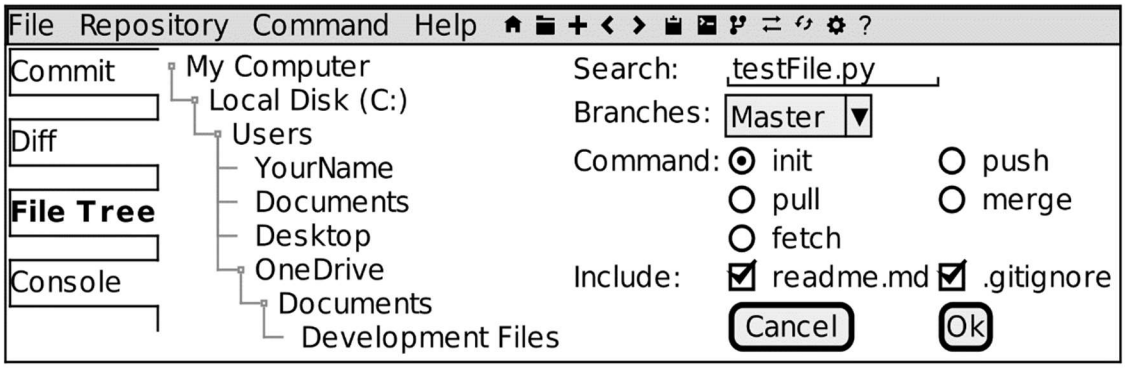

Figure 8. Application output with four menu options, vertical tabs, and a toolbar

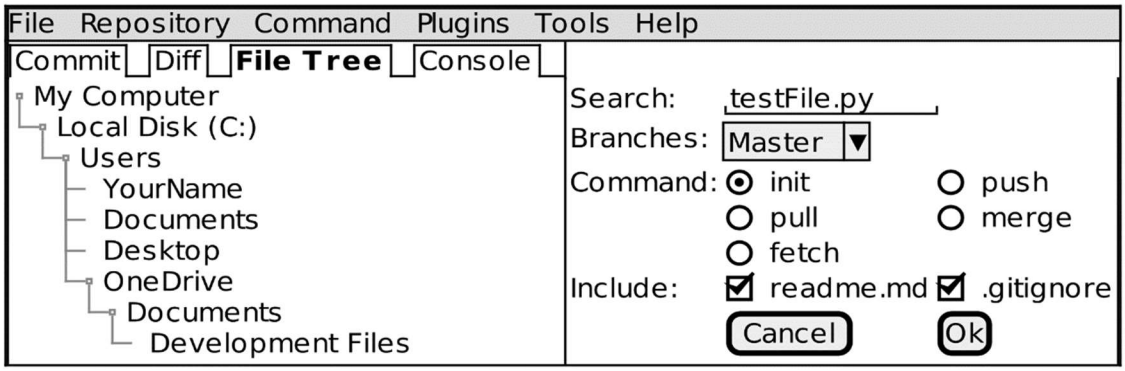

Figure 9. Application output with six menu options, horizontal tabs, and no toolbar

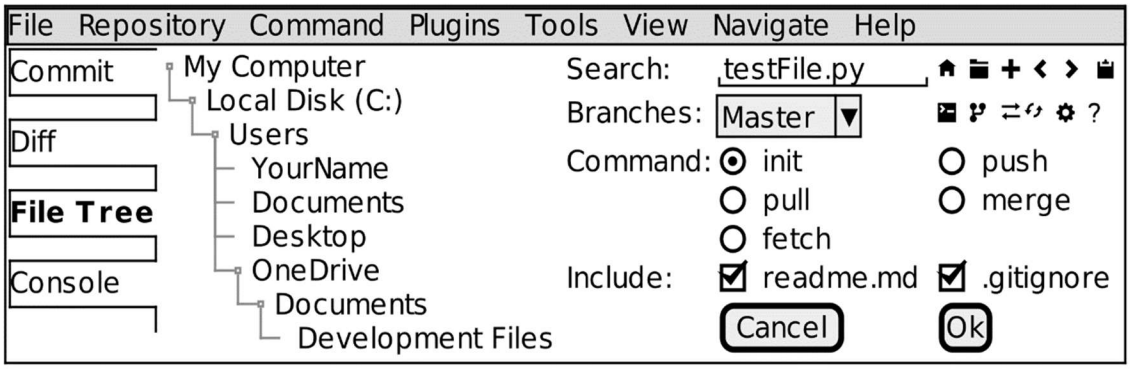

Figure 10. Application output with eight menu options, vertical tabs, and a toolbar

Database Structure | Browse Data | Edit Pra

Table: USERS

	id	email	pin
	Filter	Filter	Filter
1	1	testemail@gmail.com	1234
2	2	scott@gmail.com	1234
3	3	mike@gmail.com	4321
4	4	stephanie@gmail.com	9123

Figure 11. User and PINs stored in the USERS table

Database Structure | Browse Data | Edit Pragmas | Execute SQL

Table: SPECIFICATIONS | Filter in any column

	userID	vTabs	hTabs	menuFile	menuRepository	menuCommand	menuTools	menuView	menuNavigate	menuPlugins	menuHelp	menuItemCount	toolbar
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	0	1	1	1	1	1	1	1	1	1	8	1
2	2	1	0	1	1	1	0	0	0	0	1	4	1
3	3	0	1	1	1	1	1	0	0	1	1	6	0
4	4	1	0	1	1	1	1	1	1	1	1	8	1

Figure 12. SPECIFICATIONS table and corresponding requirements for each user

plant-uml.txt - Notepad

File Edit Format View Help

```
@startsalt
scale 5
{+
{* File | Repository | Command | Plugins | Tools | View | Navigate | Help | }
{
{/
Commit
Diff
<b>File Tree</b>
Console
} |
{T
+My Computer
++ Local Disk (C:)
+++ Users
++++ YourName
++++ Documents
++++ Desktop
++++ OneDrive
+++++ Documents
++++++ Development Files} |
{ Search: | "testFile.py" | <&home> <&folder> <&plus> <&chevron-left> <&chevron-right> <&clipboard>
Branches: | ^Master^  | <&terminal> <&fork> <&transfer><&loop-circular> <&cog> <&question-mark>
Command:  | (X) init | () push
|.| () pull | () merge
|.| () fetch | *
Include: | [X] readme.md | [X] .gitignore
|.|[Cancel] |[Ok]}
}
}
@endsalt
```

Figure 13. PlantUML output file for creating specifications image

Option #1: Mutually Conflicting Requirements

Kim *et al.* (2007) define requirements conflicts as "interactions and dependencies between requirements that can lead to negative or undesired operation of the system" (p. 419). Pressman and Maxim (2020) write that one of the most challenging tasks of being a software engineer is managing conflicting requirements. After all, several studies indicate that requirements conflicts constitute a significant factor in software projects exceeding cost and time budgets (Aldekhail *et al.*, 2016). Additionally, Baltes and Diehl (2018) indicate that expert developers need to be excellent listeners during requirements gathering and determine clients' needs when clients "can only say what they think they want" (p. 194).

For his second *Critical Thinking* assignment in *CSC505: Principles of Software Development,* the student developed a process pattern that addresses mutually conflicting requirements. Specifically, the student drafted a communication diagram for a fictitious Git application that prompts a user to specify requirements, outputting a PNG image that depicts the graphical user interface (GUI) the user specified. Also, the communication diagram addresses the problem of conflicting requirements and suggests an effective solution to the phenomenon. Pressman and Maxim (2020) write that developers often handle conflicting requirements through negotiation, asking users to rank and discuss their requirements in priority. However, such a process is not feasible to implement in an automated software solution since negotiation requires communication with stakeholders. What if two stakeholders rank a requirement as equally important yet disagree on how to implement it? If two users prioritize requirements as equally important yet disagree on their implementation, how can an application resolve such discrepancies?

This author could only think of one feasible solution: to implement a coin flip to settle disputes between conflicting requirements with two alternatives and a die roll to settle disputes between conflicting requirements with three or more alternatives (e.g., four, six, or eight menu items). Because the application stores all users' choices in its database, it checks its database for conflicting requirements after inserting a user's specifications.

If conflicting requirements are found between two alternatives, the application simulates a coin flip. If the coin lands on "heads," the current user is determined the winner, and the program outputs the user interface according to the user's preferences. If, however, the coin lands on "tails," the application discards the user's preferences, outputting an interface based on the alternative requirements. Notably, the user's updated preferences are not written to the database and are only output by the application. That way, the requirement comparison process is still based on each user's initial preferences, which users can log in and change at any time. Figure 2 shows the application's output when using a coin flip to settle conflicting requirements disputes.

In the scenario where the requirements alternatives exceed two, such as in the case of four, six, or eight menu items, the application simulates a three-sided die roll. It does so by generating a pseudorandom number between one and three, determining the specification to implement. This process can be scaled if more user choices arise. Figure 3 shows the application's output when simulating a three-sided die roll. Figure 1 shows the communication diagram containing eight classes in addition to the user interface: (a) controller class, (b) user class, (c) database class, (d) menu class, (e) software class, (f) file class, (g) coin class, and (h) dice class.

As noted, the program stores all user information in its database, including each user's login, PIN, and software requirements. The student expanded the code he wrote in the previous module, converting his code to object-oriented syntax containing classes, getters, and setters. The advantage of having users log in before specifying their requirements is that the program can store each user's requirements in its database, validate each user's identity, and check for and resolve the occurrence of conflicting requirements. Figure 4 shows a new user registering for the application. Figure 5 shows the same user logging in after entering their PIN.

Upon login, the program presents the software specification questions shown in Figure 6: (a) "How many menu options should the program contain?", (b) "Should the program contain horizontal or vertical tabs?" and (c) "Should the program contain a toolbar?" After the user enters their responses, the program writes them to its database, checks for and resolves conflicting requirements, and stores the user's specifications in a TXT file inside a folder corresponding to the user's email address. The TXT file contains the PlantUML source code to produce the GUI image corresponding to the user's specifications.

The application reads the TXT file and produces a GUI image, which it then opens for viewing. Figures 7 – 10 display four different user interfaces containing subtle variations based on different user inputs. Figures 11 – 12 show the user information stored in the program's database. All fields are binary fields indicating the user's chosen responses, except for the field titled *menuItemCount*, an integer indicating the number of fields the menu should contain. Finally, Figure 13 shows the PlantUML source code to generate a GUI image.

In conclusion, Aldekhail *et al.* (2016) analyzed and compared 20 works on requirements conflicts, dividing the work into three categories to detect and resolve conflicting requirements: (a) manual techniques, (b) automatic techniques, and (c) general frameworks. Most works

employed a manual technique, such as stakeholders and engineers discussing requirements, leading to increased effort, time, and costs. However, the student's software employed an automatic, scalable technique to manage conflicting requirements and solve disputes between stakeholders, outputting each user's specifications in graphical format and storing all responses in its database. Because the software uses an object-oriented approach, its components are easily modifiable to scale to other projects and requirements—for instance, outputting different UML diagrams such as activity or sequence diagrams rather than GUI images based on the needs of the project.

References

Aldekhail, M., Chikh, A., & Ziani, D. (2016). Software Requirements Conflict Identification: Review and Recommendations. *International Journal of Advanced Computer Science and Applications*, *7*(10). https://doi.org/10.14569/IJACSA.2016.071044

Baltes, S., & Diehl, S. (2018). Towards a theory of software development expertise. *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 187–200. https://doi.org/10.1145/3236024.3236061

Kim, M., Park, S., Sugumaran, V., & Yang, H. (2007). Managing requirements conflicts in software product lines: A goal and scenario-based approach. *Data & Knowledge Engineering*, *61*(3), 417–432. https://doi.org/10.1016/j.datak.2006.06.009

Pressman, R. S., & Maxim, B. R. (2020). *Software engineering: A practitioner's approach* (Ninth edition). McGraw-Hill Education.