1 Appendix: Mapping Program Data to Activities

In this section, we describe the algorithm we developed to map the program data, collected by our monitoring application, to an activity category. An overview of the mapping process is given in Section 4.2. The algorithm, written in C# can be downloaded from this website: `www.ifi.uzh.ch/seal/people/meyer/productive-workday`.

Data Preparation

Before we started the mapping, we added the data sets from each participant into one single database table. The table contains a list of program switches performed by participants. Each entry contains the program name, the window title, and some temporal data. Table 1 shows some example program switch entries. Subsequent to this merging and cleaning phase, the following algorithm iterated through each item in the database and mapped it to an activity category.

Base Mapping Algorithm

The basis of the mapping algorithm are lists of keywords we manually created for each activity category based on the purpose each program is used for. These lists contain program names which can distinctively be mapped to a certain activity (e.g. SourceTree is mapped to *Version Control*). In case a program name could be mapped to multiple activity categories, we also looked at the window title of each entry and usually found unique keywords, which could distinctively be mapped to one of the categories. For example, in case the program name was "Outlook" and the window title contained "calendar", "meeting", "appointment" or "task" it was mapped to *Planning*, otherwise to *Email*.

While running, the algorithm marked entries which could not be distinctively mapped to an activity category. We manually mapped these entries to a category, by extending the lists with keywords and re-running the algorithm. After each run, we manually checked the results of the new mappings, refined some keywords in rare occasions, and continued this process until we found no more errors. This approach was used to map the majority of the programs used into the following categories: *Coding*, *Debugging*, *Code Reviewing*, *Version Control*, *Email*, *Planning*, and *Reading or Writing Documents*.

Most participants did not only work on their local machine, but also on a virtual machine, remote desktop or second device. In case the participant also installed the monitoring application on a secondary machine, which most participants did, we could merge the activity from both machines and analyze the participant's activity from multiple devices. In case a participant accessed a machine where the tool was not installed, there was no way of determining the exact activity category, which is why we categorized these entries as *OtherRdp*.

Special Cases

The mapping of web browsing activity was a special case, where our algorithm differs from the standard procedure described above. As web browsers are often not only used for information retrieval, but also for emailing, instant messaging, code editing or planning, amongst others, we changed the mapping procedure for all items with a web browser as the application. First, we run the mapping algorithm from above on all web browser items that mapped many web pages to their corresponding category. With this step a database entry with the window title “outlook.com - inbox” was mapped to the category *Email*, while “github.com” was mapped to *Version Control*. We then further categorized all remaining, i.e. unmapped, items to *Work Related Browsing* or *Work Unrelated Browsing*, by manually selecting additional keywords. For example, when the window title contained words like “news”, “comic”, “cinema” or “twitter”, we assumed it to be a *Work Unrelated Browsing* session. Contrary, if the window title contained words like “developer”, “stackoverflow”, “documentation”, “javascript”, we mapped it to *Work Related Browsing*. All these mappings were again manually checked and refined as described above.

Another special case were editor apps (e.g., Notepad++ or Sublime Text) that we mapped to *Coding* in case the window title contained a document name with a file extension of a code file that could distinctively be mapped, such as “converter.cs” or “person.js”. All remaining editor entries were mapped manually based on the context given by the window title, often to *Reading or Writing Documents*, or *Planning*.

We also manually checked all automated mappings to *Debugging*, *Code Reviews and Version Control. In case such an entry could not distinctively be mapped to one of these categories, we mapped it to Coding. As a developer might also be using the browser or an editor to debug software, we defined the activity category Debugging to just contain debugging performed in the IDE.*

The remaining entries were grouped into the activity category *Other*, which aggregates several sub-categories, such as changing music, updating software, or navigating the file explorer. We also mapped programs which could not be mapped to any other activity with full certainty to *Other*.

Finally, we mapped entries marked with ‘idle’, by looking at the participants’ self-reported activities and tasks, as they often reported planned and informal meetings, breaks and lunches, amongst other activities. This information could in many cases be used to infer if ‘idle’ time entries belong to one of these categories. In cases where the participant did not report a meeting or break, we had no way of identifying the intent of ‘idle’ time, which is why the amount of time spent with planned and unplanned meetings might be higher than reported. We also categorized the use of instant messaging tools (e.g. Skype, Skype for Business, Google Hangout) to *Planned Meeting* in case the participant self-reported it as planned, otherwise we mapped them to *Informal Meeting*.

Table 1: Excerpt from the Program Switches Table, the Basis for the Mapping Algorithm.

Id	Participant	Process	Window Title
1	S01	outlook	Inbox - [obfuscated email address] - Outlook
2	S01	idle	idle
3	S01	taskmgr	Windows Task Manager
4	S01	firefox	C# - How to get all public calendar of specific user? - Stack Overflow
5	S01	devenv	[obfuscated project name] - Microsoft Visual Studio
6	S02	iexplore	IBM Rational ClearQuest - Windows Internet Explorer
...			

Validation

The implementation and iterative refinement of the mapping algorithm was created by the first author of the paper, and validated by another researcher from the author's research group. The second researcher validated the mapping by examining the first few hundred mappings for each activity category and debugging through the mapping algorithm implementation. The validation resulted in only two minor changes, where two web pages were mapped to *Work Unrelated Web Browsing* instead of *Work Related Browsing.*