Abstract
Today many end-users implement software tools on their own due to their high, general computer literacy and the increased user-friendliness of the software. The most popular and ubiquitous software packages to implement such personal productivity tools for analytical purposes are spreadsheets. A general problem is that a typical knowledge worker in the role of an end-user does not have explicit training in software engineering. Therefore, in this article some aspects of software engineering in general and spreadsheet engineering in particular are discussed. Based on this discussion several guidelines for implementing spreadsheet-based tools are given. Applying these guidelines helps to design spreadsheet-based tools that can be easily reused, extended, and shared with other users. A case study illustrates how to use Microsoft Excel to implement a tool which is in compliance with these guidelines. The case study is designed as an introduction for MBA students on how to implement spreadsheet-based analytical tools.

1. Introduction
In all types of business, people have to accomplish tasks every day. From an information processing point of view, completing a task involves steps like identifying the relevant issues, getting the necessary data, turning them into meaningful information, anticipating possible outcomes, making a decision, and implementing the decision. With the ubiquitous availability of computers, today task completion is often strongly supported by software applications. The applications used by a knowledge worker range from large, standardized, organization-wide enterprise applications to small software tools which are specifically adapted to the requirements of the task and the user. These individual applications are commonly referred to as personal productivity tools.

In the following section, the discussion focuses on analytical tools, i.e., a special category of personal productivity tools which are intended to support quantitative analyses. Knowledge workers often implement and maintain such tools themselves. As end-users, knowledge workers are generally experts in a particular domain (e.g., operations management), but problems may arise because they do not have explicit training in software engineering.

In the next section, software engineering is discussed from the perspective of accomplishing of a business task. The discussion focuses in particular on spreadsheet engineering of analytical tools because this approach is commonly used by knowledge workers (e.g. Chan and Storey, 1996 and Leon et al., 1996).

In the third section a case study shows how to implement a spreadsheet-based analytical tool. The target audience is MBA students who have only a little experience in the development of software. Nevertheless, a basic knowledge of spreadsheet packages is assumed (for an introduction, see for example Grauer and Barber, 2002).

2. Software Engineering of Analytical Tools
When deciding to implement a personal productivity tool, the knowledge worker typically has the main goal of increasing his or her performance in completing a specific task. When looking at the process of completing a task, different types of activities can be identified. Figure 1 outlines a typical sequence of activities (adapted from Anderson et al., 2000).
The origin of many tasks is that the management becomes aware of opportunities or risks for the business. The next activity, which includes defining the task, formulating the objectives, getting the relevant data, and determining restrictions for possible courses of action, structures the issues. After this structuring, in the stage of analysis, feasible alternatives are designed and evaluated. This analysis can take two basic forms: qualitative and quantitative (see Anderson et al., 2000). A qualitative analysis mainly relies on the experience and intuition of the knowledge worker. In contrast, a quantitative analysis consists of a mathematical representation of the issue which can be evaluated by a computer-based system. The knowledge worker must decide which aspects should be analyzed qualitatively and which should be analyzed quantitatively. For complex tasks this decision often gets more difficult because several analyses must be made. Next the insight gathered by the analyses must be synthesized and a decision must be made. Finally this decision is implemented by some action. It should be added that these activities not only occur in the presented sequence but in general are more or less interwoven.

If the quantitative analysis is to be supported by a computer-based system, then the analytical tool must of course be produced first. The discipline which is concerned with software production in general is called software engineering. Software engineering has the goal of delivering high-quality software at agreed cost and on schedule (Humphrey, 1997). In order to reach this goal, various generic software engineering process models have been developed (see Sommerville, 2000 for a comprehensive treatment). These models can be classified into two broad classes: requirement-oriented and reuse-oriented approaches.

Two almost contrary, requirement-oriented approaches are the "waterfall model" and the "evolutionary model". If the software can be specified in detail, then the waterfall model with separate and distinct phases of specification and development can be used. In contrast, when using the evolutionary approach, the specification and development are interleaved. In the evolutionary approach, an initial prototype is developed and evaluated. Based on this evaluation, an improved version is implemented and evaluated again. This procedure is repeated until the final application with the desired features is implemented. Due to the fact that a system with limited functionality is available very quickly, this approach is also called "rapid application development" (see Bocij et al., 2003). Table 1 summarizes the most important features of two approaches presented.

<table>
<thead>
<tr>
<th>Approach</th>
<th>Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>Waterfall Model</td>
<td>Minimizes planning overhead.</td>
</tr>
<tr>
<td></td>
<td>Minimizes wasted efforts.</td>
</tr>
<tr>
<td></td>
<td>Addressing mistakes made in previous phases is difficult.</td>
</tr>
<tr>
<td>Evolutionary Model</td>
<td>Can be used if no one can fully specify the requirements in advance.</td>
</tr>
<tr>
<td></td>
<td>Adaptation to changing requirements is possible.</td>
</tr>
<tr>
<td></td>
<td>Steady progress is visible.</td>
</tr>
<tr>
<td></td>
<td>Number of necessary iterations is unknown.</td>
</tr>
</tbody>
</table>

Besides such requirement-oriented approaches, reuse-oriented process models are becoming increasingly important (see Sommerville, 2000). The basic idea is to assemble a new application from existing software components by configuring and combining them in a new way. More recent programming languages support this concept by providing advanced concepts for the modularization of the functionality (e.g., object-
oriented programming with Java and JavaBeans\(^{(1)}\). In the context of analytical tools, Savage (1997) coined the term "decision object" for software tools which an end-user can interactively use to solve a specific problem. Using this approach, the completion of a more complicated task consists of the intelligent combination of such elementary decision objects by the knowledge worker. Having modularized a software application into components also allows distributed processing on several computers. A recent technological development for distributed processing are web services, i.e., small, reusable tools which receive and transmit data over the Internet using XML as the encoding scheme (see, e.g., the introduction from Microsoft\(^{(2)}\)).

The software engineering of an analytical tool consists of three main phases: formalization, design, and coding (e.g., Humphrey, 1997). In order to make the task accessible for a software application, some kind of formalization must be made. For analytical tools, Hesse (1997) proposes for this phase first to picture and paraphrase the task, then to construct a verbal model, and finally to construct a mathematical model.

While during formalization the question "what is the functionality" is answered, the design phase deals with "how should the functionality be provided". Humphrey (1997) surveys the three common ways of representing a design: graphically, with pseudo code, or mathematically. Janvrin and Morrison (2000) introduce a graphical representation for the design of spreadsheets similar to data flow diagrams. When applying general software engineering techniques like the Unified Modeling Language (UML), a software application can be both formalized and designed using a coherent set of representation techniques (for a comprehensive treatment of UML see Booch, 1999 or the website of the Object Management Group\(^{(3)}\)).

Finally, in the coding phase the design is implemented using a software development environment. If an end-user implements an analytical tool, then often a spreadsheet package which integrates the development environment is used. Grossman (2002) provides a comprehensive spreadsheet engineering research framework. The literature contains various recommendations on how to implement advanced spreadsheet tools. Conway and Ragsdale (1997) provide guidelines which should help with implementing reliable, auditable, and modifiable spreadsheet tools. Edwards et al. (2000) provide numerous "best practice points" for a spreadsheet implementation. Ragsdale (2003a, ch. 3.7) and Raffensperger (2001) formulated several rules for the layout of worksheets. Albright (2001, ch. 14.2) provides guidelines for implementations that use the programming language "Visual Basic for Applications" (VBA), which is integrated in the spreadsheet package Excel. In addition to this literature which explicitly addresses the implementation, various OR/MS-related textbooks implicitly show how to implement spreadsheet tools by providing best practice examples, like Albright (2001), Anderson et al. (2000), Camm and Evans (2000), Denardo (2002), Hesse (1997), Hillier et al. (2000), Kirkwood (1997), Lawrence and Pasternack (2002), Moore and Weatherford (2001), Ragsdale (2003a), Savage (2003), Winston and Albright (2001).

In order to guide the implementation of an analytical tool, I propose the five general guidelines shown in Table 2. The first guideline emphasizes that the tool can be used if the same or a similar task (re)occurs. Often an end-user implements an analytical tool according to an evolutionary software engineering process. Following this approach, first a basic version of the tool is implemented. If it seems promising that the task accomplishment can be further improved, then an extended version is created. This extension of the tool continues until the end-user perceives no more "value-added" of more functionality or the technological restrictions of the spreadsheet are met (see e.g. Savage, 1997).

In order to guide the implementation of an analytical tool, I propose the five general guidelines shown in Table 2. The first guideline emphasizes that the tool can be used if the same or a similar task (re)occurs. This feature is crucial for an analytical tool because it enables the knowledge worker to gradually develop a set of personal "decision objects" which help in achieving a higher performance in completing tasks. The second guideline is intended to simplify using the tool for "what-if" questions. The other three guidelines cover aspects of the typical input/processing/output-scheme of data processing.

---

\(^{(1)}\) http://ite.pubs.informs.org/Vol4No1/Scheubrein/index.php#JavaBeans

\(^{(2)}\) http://ite.pubs.informs.org/Vol4No1/Scheubrein/index.php#MicrosoftWebServices

\(^{(3)}\) http://ite.pubs.informs.org/Vol4No1/Scheubrein/index.php#OMG
Guidelines for Implementing an Analytical Tool

A The tool should be reusable and extensible.
B One should be able to change parameters easily in order to evaluate different scenarios.
C The data input should be user-friendly.
D The processing logic should be appropriately documented in order to allow understanding and modifying of the tool later.
E The result of the analysis should be presented in a way to help in gaining insight.

End-users who have some experience in implementing spreadsheet tools are likely to develop personal styles of software development which will be evident even in the first version of a tool. Nevertheless the guidelines from this article or other sources (see above) might give hints on how to further improve their personal style of software development.

In the following section, it is shown how the guidelines in Table 2 translate to the implementation of an analytical tool in a spreadsheet environment. The final version of the tool can be implemented in less than one hour by following the instructions. Alternatively, the tool can be downloaded in the various stages of development.

3. A Case Study on Spreadsheet Engineering

In this case study we develop a forecasting tool step by step. In order to concentrate on the software engineering aspects, a simple exponential smoothing method is implemented. Nevertheless, the final tool has many essential features of a larger application.


Incorporating concepts from that literature, the course at the University of Hohenheim is designed for MBA students with a major in operations and logistics management. These students are familiar with the basic functionality of spreadsheets but have no particular knowledge of software engineering. The course starts with the following tasks for the students:

1. Give a short presentation of a real-world spreadsheet application. The intention is to motivate the students and show them that the content of the course is relevant and that spreadsheet packages are really general-purpose tools.
2. Go through the "Excel Tutorial" of Albright (download from Indiana University’s website or get the local copy). This tutorial is a Word document with embedded Excel spreadsheets. It provides an excellent, compact overview of many useful Excel features. The purpose of this homework is to refresh the students’ knowledge. In order to check this homework, some students can be asked to give a short presentation of a function which was new to them.
3. In class it is shown how to implement the forecasting tool step by step. The students are given as a homework assignment the task of building a tool with comparable complexity (e.g., a tool to generate a Boston Consulting Group portfolio, a tool to analyze the life cycle of a product, an A-B-C analysis of inventory, or a tool implementing the basic economic order quantity model). All tools have to be uploaded to a Yahoo Group. The purpose of using such an Internet community platform is that all students should have access to all tools in order to support the effect of learning from each other in the class.

In the following sections, the forecasting tool is implemented in Microsoft Excel 2002. We use only standard spreadsheet features, so the tool could also be built using previous versions of Excel or other spreadsheet packages. A zip-compressed workbook and a video illustrate each development stage. The video is encod-
ed using Microsoft Media 9 series technology. In case of problems viewing a video, the latest version of this technology can be downloaded from the Windows Media homepage(7)

3.1. Task Requirements
This case assumes that the demand of a product has to be forecasted. When looking at the time series of the demand in Table 3 (data adapted from Stevenson, 2002), it can be concluded that the demand is basically stable.

<table>
<thead>
<tr>
<th>Period t</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>Demand y(t)</td>
<td>4200</td>
<td>4000</td>
<td>4300</td>
<td>4000</td>
<td>4100</td>
<td>3900</td>
<td>4600</td>
<td>4400</td>
<td>4500</td>
<td>3800</td>
<td>4000</td>
</tr>
</tbody>
</table>

A forecast method for stable demand is exponential smoothing (e.g. Winston and Albright, 2001, ch. 16.8). In exponential smoothing, the forecast for the next period is based on the time series of the demands of the past. For the next period t+1, the forecast p(t+1) for the demand y(t+1) is recursively defined on the demand y(t) in the current period and its forecast p(t). The smoothing parameter α is used to compute a weighted sum of the forecast and the actual demand of the current period as specified in equation (1). The parameter α has to be set in the range from 0 to 1.

\[ p(t + 1) = (1 - a) \cdot p(t) + a \cdot y(t) \]  

(1)

3.2. Step 1: Getting the Result
When setting the parameter α to the value 0.46, the time series and the formula can easily be entered into a worksheet (see Figure 2(8)). In Column B there is the period, in Column C the demand, and in Column D the formula with the forecast.

3.3. Step 2: Isolating Parameters
While this worksheet now answers the question of which is the forecast for the next period (see cell D16 in Figure 2), all of the guidelines given in Table 2 have been ignored. In particular, it is not easy to change the smoothing parameter α because it is "hard-wired" into all the formulas in Column D. Therefore, the first step is to move the parameter to a separate cell (F5). With this change, now Guideline B is achieved better in the workbook shown in Figure 3(9).

Figure 2: Forecasting the demand
Figure 3: Separating the parameter

(7) http://www.microsoft.com/windows/windowsmedia/
(8) http://ite.pubs.informs.org/Vol4No1/Scheubrein/Book1.zip
(9) http://ite.pubs.informs.org/Vol4No1/Scheubrein/Book2.zip
3.4. Step 3: Using Named Cells
If the cell F5 is additionally given a name, then the readability of the formula increases, and this results in a better achievement of Guideline D. In order to name a cell, it has to be selected, and the name can be typed into the "name box" at the left end of the formula bar (see Figure 4\(^{(10)}\)). Defined names can later be modified by using the menu entry <Insert><Name><Define...>. In this example, the name "alpha" is given to cell F5. The formula in Column D must be modified to reference this name. As shown on the right side of Figure 4, the formulas in the worksheet now look much more similar to Equation (1).

Excel provides various possibilities for formatting a cell under the menu entry <Format><Cells...> (e.g., font, border, background pattern, and color). While it is not too important which format is used, it is helpful to apply the same formatting to all applications in order to document the worksheet. Using the same coding scheme also helps others who are using several applications by the same author to understand how the software works.

As shown in Figure 5\(^{(11)}\), the cells with the input data, the parameter, and the result can be marked, for example, with a different background color. Intermediate results are not especially emphasized. After selecting a cell, its background color can be changed by clicking on the icon "Fill Color" and selecting a color from the palette.

Also in this step a title is added to the worksheet, and the table is formatted. The general policy of adding such elements is that they should be informative. In contrast, adding decorative elements is more a question of personal style.

3.5. Step 4: Clarifying the Worksheet Layout
When developing a more complicated application, it becomes difficult to distinguish input and output. Following Guideline D, it is helpful to format the cells according to what they contain. In general, the contents of a cell can be categorized as follows:

- **The input data** is fixed for the considered analysis.
- **The parameters** are under the control of the user. Parameters are typically called decision variables in optimization models.
- The **results** are computed by a method based on the input data and the parameters.
- **Intermediate results** reflect the necessary steps to compute the final result. They are used to reduce the length of a formula in a single cell and to provide an opportunity to find errors in the computation.

As shown in Figure 5, the cells with the input data, the parameter, and the result can be marked, for example, with a different background color. Intermediate results are not especially emphasized. After selecting a cell, its background color can be changed by clicking on the icon "Fill Color" and selecting a color from the palette.

Also in this step a title is added to the worksheet, and the table is formatted. The general policy of adding such elements is that they should be informative. In contrast, adding decorative elements is more a question of personal style.

3.6. Step 5: Enhancing the Input of Data
While the user can now easily identify which cells are intended to hold the input, the process of entering the actual data can be enhanced with the addition of some error-checking (Guideline C). For this purpose, Excel provides a function for validating input data. In order to activate this validation for a cell, it must selected first. Then the validation rule can be specified in the menu entry <Data><Validation...>. Figure 6\(^{(12)}\) shows how to specify that the user has to input a number in

---

\(^{(10)}\) [http://ite.pubs.informs.org/Vol4No1/Scheubrein/Book3.zip]
\(^{(11)}\) [http://ite.pubs.informs.org/Vol4No1/Scheubrein/Book4.zip]
\(^{(12)}\) [http://ite.pubs.informs.org/Vol4No1/Scheubrein/Book5.zip]
the range from 3000 to 5000. While the requirement that the input has to be a number is caused by the method, the constraint on the range could stem from a business rule. For example, this could reflect the rule that the demand should only be forecasted with the method if it is in the normal range from 3000 to 5000. If the demand is outside this range, then a more detailed analysis should be made.

3.7. Step 6: Enhancing the Input of Parameters
Also following Guideline C, the way of changing the parameter stored in the cell named "alpha" can be made more user-friendly. In Excel, a scroll bar can be placed on the worksheet and linked to the parameter cell. The scroll bar is available from the toolbar "Control Toolbox" (see Figure 7). If this toolbar is not visible, it can be activated by selecting the menu entry <View><Toolbars><Control Toolbox>. After clicking on the scroll bar icon on the control toolbox, the user can draw the outline of the scroll bar on the worksheet. By clicking with the right button on the placed scroll bar, its properties can be changed (see right side of Figure 7). By default, a scroll bar places an integer value between 0 (property "Min") and 32767 (property "Max") in the cell defined by the property "LinkedCell".

While the scroll bar only provides integer values, the smoothing parameter \( \alpha \) must be a real value in the range from 0 to 1. Therefore the additional cell F6, which holds the current value of the scroll bar, is used. The property "linked cell" should be set to F6 and the property "Max" should be changed to "100". Finally the formula in the parameter cell has to be changed to "=F6/100" to reflect the position of the scroll bar. After having exited the design mode, the smoothing parameter can now be changed using the mouse and no invalid value can be entered anymore (see Figure 8). (13)

3.8. Step 7: Gaining Insight
In order to meet Guideline E, a chart is often useful for supporting an analysis. In this case, a line chart is appropriate for visualizing the time series and the resulting forecast. In order to include this chart, first select the range from B5 to D16. Then click on the icon "Chart Wizard" and place a line chart using basically the default settings on the worksheet (see Figure 9). (14)

---

(13) http://ite.pubs.informs.org/Vol4No1/Scheubrein/Book6.zip
(14) http://ite.pubs.informs.org/Vol4No1/Scheubrein/Book7.zip
If the smoothing parameter is now changed by moving the scroll bar, the effect is instantly visible in the chart.

Another facet of extensibility is that a user is provided with some kind of documentation about the current functionality of the tool. This documentation becomes especially crucial if the tool is only used every once in a while. Such a basic documentation can be included in the worksheet in several ways. First, Excel offers the possibility of documenting a single cell by right-clicking and choosing the entry "Insert Comment" from the context menu. Now the cell has a small, red triangle in the upper right corner. If the user moves the cursor over the cell, the comment opens (see cell D6 in Figure 11). A second possibility is to draw a text box on the worksheet. For this, first select <View><Tool-bars><Drawing> to open the drawing bar, use <Auto-shapes><Callouts> to draw the box, and then write the description in the box.
Additionally, it is useful to place two buttons called "Start" and "Example" on the page (see Figure 12\(^{(17)}\)). The "Start" button provides a dialog in which the user can enter new data. Using the "Example" button automatically inserts some typical data, in order to demonstrate the functionality of the tool to a new user.

These procedures can be added using the macro language of the spreadsheet package. In Excel this language is called VBA. While a detailed discussion of the program in the workbook of this step is beyond the scope of this article, you can download it and switch to the editor using the key combination ALT-F11. It should be noted that for security reasons, the execution of macros is by default deactivated in more recent versions of Excel. In order to activate it, the menu entry <Tools><Macro><Security...> has to be selected, and the security level should be set to "medium".

For people new to programming, the macro recorder can be used to provide the skeleton of the procedures. After having started the macro recorder from the menu entry <Tools><Macro><Record New Macro...>, the user manually inputs the steps which are necessary, and the corresponding procedure is generated automatically. This approach to programming is normally only successful for very small procedures. Normally, the recorded program has to be revised in the visual basic editor of Excel. Such a revision requires sufficient experience in programming VBA. Coming from an OR/MS background, especially Albright (2001) can be recommended to learn VBA, because he shows which of the numerous constructs are essential for implementing decision support systems.

4. Summary

In this article, some aspects of the software engineering of personal productivity tools are discussed. Often these tools are implemented by an end-user using a spreadsheet package because this type of software is easily accessible and provides a broad range of functionality.

Nevertheless, implementing a spreadsheet tool is not a straightforward activity. Therefore, this paper provides guidelines for end-users on how to build an analytical tool. A case study illustrates how these guidelines translate into the implementation of a spreadsheet-based tool which can be reused and extended even though only basic functions of the spreadsheet package are used.

Acknowledgements

The author would like to thank Erhan Erkut for providing a workshop in the year 2000 on how to use spreadsheet software as a didactic instrument. Also, a lecture from John Lawrence in the year 2001 provided valuable additional insights. At the INFORMS conference in November 2002, Ronald Lodewyck gave a useful presentation on how to produce and use videos for teaching purposes. Valuable suggestions based on an early version of this paper were made by Thomas A. Grossman. Last but not least, an associate editor of ITE and four anonymous referees helped improve this document with their comments.
References


MICROSOFT (2003), "Defining the Basic Elements of .NET", ELEMENTS OF END-USER SOFTWARE ENGINEERING

OBJECT MANAGEMENT GROUP (2003), http://www.omg.org/


