Design of a Graphical User Interface and Specific Widgets for a Dashboard for Monitoring QS-Metrics from SYNECT Test Management

Bachelor’s Thesis
Submitted in Partial Fulfillment of the Requirements for the Degree of Bachelor of Science

by

OLGA MIKHAYLOVA CHERNETSKY
Thesis Supervisor: Prof. Dr. Gerd Szwillus

Paderborn, December 2016
Declaration
I hereby declare that I am the sole author of this bachelor thesis and that I have not used any sources other than those listed in the bibliography and identified as references. I further declare that I have not submitted this thesis at any other institution in order to obtain a degree.

Original Declaration Text in German:

Erklärung
Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen worden ist. Alle Ausführungen, die wörtlich oder sinngemäß übernommen worden sind, sind als solche gekennzeichnet.

__________________________________________  ________________________________
City, Date                                      Signature
# Contents

1 Introduction and goals  
  1.1 Motivation ........................................... 1  
  1.2 Structure ............................................. 2  

2 Fundamentals ........................................... 3  
  2.1 Dashboard ............................................. 3  
    2.1.1 Common mistakes .................................. 4  
    2.1.2 Fundamental considerations ....................... 5  
    2.1.3 Visual Perception fundamentals ................. 6  
    2.1.4 Key goals .......................................... 9  
    2.1.5 Graphs ........................................... 10  
    2.1.6 Other display media ............................... 14  
    2.1.7 Critical design practices ....................... 14  
  2.2 Contextual Design ................................... 16  
  2.3 Scrum ................................................. 17  

3 Context Analysis ....................................... 19  
  3.1 Work modeling ...................................... 20  
  3.2 Current solution flaws ............................... 24  
  3.3 User requirements ................................... 26  
  3.4 Conclusions .......................................... 27  

4 Proposed Solution ..................................... 29  

5 Summary and Future Work ............................... 35  
  5.1 Summary ............................................. 35  
  5.2 Future Work .......................................... 36  

Bibliography .............................................. 37
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Proximity: Objects placed close to one another on a 2-D space are perceived</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>as belonging together.</td>
<td></td>
</tr>
<tr>
<td>2.2</td>
<td>Similarity: Objects that look similar to one another scattered among other,</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>different elements are perceived as a group.</td>
<td></td>
</tr>
<tr>
<td>2.3</td>
<td>Enclosure: Surrounding a set of objects with a frame (enclosure) is another</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>way to separate them from the rest.</td>
<td></td>
</tr>
<tr>
<td>2.4</td>
<td>Closure: The principle of closure means viewers tend to see the whole picture</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>when it is not completely visible.</td>
<td></td>
</tr>
<tr>
<td>2.5</td>
<td>Continuity: Objects that are aligned with one another are perceived as a</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>whole.</td>
<td></td>
</tr>
<tr>
<td>2.6</td>
<td>Connection: Objects connected to one another are also perceived as part of</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>the same group.</td>
<td></td>
</tr>
<tr>
<td>2.7</td>
<td>Different regions of the screen are associated with different emphasis levels.</td>
<td>10</td>
</tr>
<tr>
<td>2.8</td>
<td>Bullet graphs are useful for expressing single key measures.</td>
<td>11</td>
</tr>
<tr>
<td>2.9</td>
<td>Bar graphs are a better choice than pie charts for displaying quantitative</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>information.</td>
<td></td>
</tr>
<tr>
<td>2.10</td>
<td>Stacked bar graph are the right choice only if we must display multiple</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>instances of a whole and its parts, with emphasis on the whole.</td>
<td></td>
</tr>
<tr>
<td>2.11</td>
<td>When the scale does not begin at zero, plots that encode values based on</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>height or area can be confusing. In this cases a dot plot may be the best</td>
<td></td>
</tr>
<tr>
<td></td>
<td>choice.</td>
<td></td>
</tr>
<tr>
<td>2.12</td>
<td>Small multiples are a tabular arrangement of similar graphs. Image source:</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>Wikipedia [Wik16a].</td>
<td></td>
</tr>
<tr>
<td>2.13</td>
<td>Sparklines provide a quick overview over the evolution of a measure in time.</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>Image source: Wikipedia [Wik16b].</td>
<td></td>
</tr>
<tr>
<td>3.1</td>
<td>SYNECT Scrum dashboard, overview</td>
<td>19</td>
</tr>
<tr>
<td>3.2</td>
<td>SYNECT Scrum dashboard, teams</td>
<td>20</td>
</tr>
<tr>
<td>3.4</td>
<td>Flow model</td>
<td>21</td>
</tr>
<tr>
<td>3.5</td>
<td>Sequence model: Developer</td>
<td>22</td>
</tr>
<tr>
<td>3.6</td>
<td>Sequence model: Tester</td>
<td>23</td>
</tr>
<tr>
<td>3.7</td>
<td>Physical model</td>
<td>23</td>
</tr>
<tr>
<td>3.8</td>
<td>Photo: The current dashboard and instructions, as seen from the coffee</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>kitchen.</td>
<td></td>
</tr>
<tr>
<td>3.9</td>
<td>Colors of the test results on the current dashboard</td>
<td>25</td>
</tr>
</tbody>
</table>
List of Figures

3.10 Instructions of the current dashboard. 26
4.1 Prototype of the proposed solution. 29
4.2 Label with date and time of the test run is shown on hovering over each test suite. 30
4.3 List of newly failed tests is shown along with the label when hovering over an alert icon. 30
4.4 Label with date and time of the test run is shown on hovering over each test suite. 31
4.5 The summary button opens an overview of all tests. 32
1 Introduction and goals

The goal of this thesis is the design of a new dashboard to assist an agile software development team at dSPACE GmbH in Paderborn, Germany. The team is responsible for SYNECT, a data-management software developed by the company, one of its leading products. The current dashboard is displayed on a shared monitor at a central position, where all team members can easily access it. It shows the results of all the different tests that were performed on the software product throughout the day. The current solution presents several flaws and is out of date, which is why the company is interested in a new design. The first part of this project consists in the analysis of the working environment, where the different user roles and which information is relevant for each one of them is determined. This data is then applied to the design process in order to achieve an optimal solution.

1.1 Motivation

Information technologies are essential to any enterprise nowadays. Even more so for a company specialized in technological solutions. Tools like information dashboards help managers and employees maintain “situation awareness” [Few13, p. 30] - know and understand at any time the information that is relevant for them in order to perform their jobs. A well-designed dashboard is one that is capable of communicating this information effectively, making appropriate use of visual perception theory. The design of such a tool for an agile software development team poses an interesting challenge: The possibility to work closely with the team is a perfect opportunity to apply Contextual Design techniques proposed by Hugh Beyer and Karen Holtzblatt [HB] in order to ensure that the end product will most closely meet the users’ needs.

The SYNECT development process is organized following the agile Scrum philosophy, meaning constant, rapid changes must be kept track of. Accordingly, the dashboard needs to show the latest results of the different tests running on the product in real time, keeping the team aware of the most urgent work to be done.
1.2 Structure

In this thesis Contextual Design methodology along with Stephen Fews guidelines for dashboard design are applied to develop a concept of a dashboard for a software development team.

In the following chapter theoretical concepts are explained: contextual design, dashboard design guidelines and Scrum (agile software development methodology). Thereafter, the performed context analysis is detailed, with models describing the users’ work practices, an overview of the current solution flaws and the users’ defined requirements for the new dashboard.

Finally, the proposed solution is explained in detail, with images of the prototype and justifications of the design.
2 Fundamentals

This thesis closely follows Stephen Few’s guidelines for dashboard design, depicted in his book “Information dashboard design: displaying data for at-a-glance monitoring” (Analytics Press, 2013) [Few13]. His work is a specialized handbook on dashboard design. It covers all relevant topics, from visual perception fundamentals to critical design practices, with plenty of graphical examples and backed up by years of experience.

The design process is user centered, based on Beyer’s and Holtzblatt’s Contextual Design techniques [HB]. It is only logical to apply a user centered approach in this context, since the design is aimed at a very specific set of users and the opportunity is given to observe and interview them in their workplace through the whole duration of this project. Beyer and Holtzblatt have developed a comprehensive user centered design methodology, which is partly applied in this work.

The techniques described by these authors are summarized in the following sections, along with a brief introduction to the Scrum software development process, which is needed to understand the work practice of the users.

2.1 Dashboard

Stephen Few defines a dashboard as “a visual display of the most important information needed to achieve one or more objectives, consolidated and arranged on a single screen so the information can be monitored at a glance” [Few13, p. 26].

It is a visual display, meaning heavy use of graphics is recommended, because they communicate more efficiently than text alone if visual perception theory is correctly taken advantage of.

It displays information needed to achieve one or more objectives, meaning the focus is set on the specific goals of the users.

It has to fit on a single screen, which means the data cannot be spread across several pages and no scrolling is allowed. Although, secondary views with detailed information can be accessed from it.

Information can be monitored at a glance, which means only an overview of the most crucial data is presented on the main screen, details made available only on specific interactions.

So, the primary goal of a dashboard is to keep the users updated on everything they need to know at any time in order to achieve their goals and nothing apart from that. Situation Awareness [Few13, p. 30] is a useful concept to understand this functionality. It is defined as perceiving and understanding what is going on
in ones environment and being able to use this knowledge to plan future actions.

2. Fundamentals

2.1.1 Common mistakes

Few provides a list of the most common bad practices in dashboard design. The currently used dashboard presents some of the flaws described in this section; it is analyzed in section 3.2. These are the most common mistakes according to Few [Few13, p. 35]:

1. *Exceeding the boundaries of a single screen.*
   Taking too much space of the screen so that scrolling is required or fragmenting the data into separate screens exceeds the purpose of a dashboard.

2. *Supplying inadequate context for the data.*
   Single values do not carry enough information - comparative measures are needed to provide a useful insight.

3. *Displaying excessive detail or precision.*
   Only high level information, which provides a quick overview, belongs on a dashboard. Navigation to details can be provided via links.

4. *Expressing measures indirectly.*
   A measure is deficient if it fails to communicate the meaning the viewer needs, for example when calculations are required from the user to get the data he is actually looking for.

5. *Choosing inappropriate display media.*
   This is one of the most common mistakes. It is explained in detail in the visual perception chapter.

6. *Introducing meaningless variety.*
   Consistency in the display media allows viewers to use the same perceptual strategy for interpreting all of the data, which saves time and energy.

7. *Using poorly designed display media.*
   The visual perception chapter provides the necessary theory for good design.

8. *Encoding quantitative data inaccurately.*
   Choosing the wrong scale can lead to misinterpretations of the data.

9. *Arranging information poorly.*
   The most important information must be prominent on the screen.

10. *Highlighting important information ineffectively or not at all.*

11. *Cluttering the display with visual effects.*
   Useless decorations distract viewers attention and waste time and space.
12. *Misusing or overusing color.*
   More about the optimal use of color can be found in the visual perception chapter.

13. *Designing an unattractive visual display.*
   Aesthetic aspects are discussed in detail in Chapter 12 “Critical design practices”.

### 2.1.2 Fundamental considerations

Few recommends to consider the following user related features to determine how a dashboard should be designed [Few13, p. 65]:

1. **Update frequency.**
   If the dashboard is updated more often than once a day, the time of the last update must be shown. A dashboard which has to be updated so often also has to be simpler, too, to not overwhelm the users with too much constantly changing information.
   For example in this project, the so-called unit tests run several times a day and new results are available every hour, so the dashboard has to be updated in real time.

2. **User expertise.**
   The complexity level must be adequate to the audience.
   For a software development team it can be safely assumed that user expertise is on an expert level.

3. **Audience size.**
   The more we can customize a dashboard, the more effective it will be. For growing number of users with different roles it becomes more difficult to meet everyones needs.
   The SYNECT crew consists of four Scrum teams, with six to ten members each. Different roles inside each team are defined (developer, tester, project owner(PO)) plus a test manager who supervises the overall testing process. The dashboard must fit the different needs of each user type without becoming too complex.

4. **Technology platform.**
   Design strategies differ for applications running on a desktop PC or mobile device from those running on a web server, constraints associated with the latter having to be taken into account.

5. **Screen type.**
   Size, viewing distance, aspect ratio and resolution must be taken into account to make sure that every element of the dashboard can be seen clearly.
6. Data type.
Two types of data can be showed on a dashboard:

- Quantitative
- Non-quantitative

Due to the nature of monitoring, quantitative information is widely dominant. The usefulness of this kind of information is highly dependent on the right context: evolution in time and comparison between key measures can provide this context, which can be also enriched by evaluative or qualitative indicators and predictions about the future.

Both data types are present on the dashboard designed in this thesis: icons with non-quantitative meaning share the screen with graphics displaying quantitative test results data.

2.1.3 Visual Perception fundamentals

Few focuses on the following aspects of visual perception as key to an optimal dashboard design:

- Limits of working memory
- Encoding data for rapid perception
- Gestalt principles of visual perception

The three fundamental types of memory are iconic memory, working memory and long-term memory. Iconic memory is the first “stage” information goes through, some so-called “preattentive processing” takes place here (a subconscious information gathering from the environment and some preliminary interpretation, which occurs at a very high speed).

Working memory is where information is consciously processed ([Few13, p. 79]). It has limited capacity: Only three or four “chunks” of visual information can be stored at a time, as was demonstrated by Nelson Cowan and J.Scott Saults ([SC07]). Through good designed graphical patterns it is possible to group more information together into one “chunk” than with text. This limited capacity of the working memory is the main reason all information must fit in one screen. (see Section 2.1.1, page 4)

It is possible to make communication even more efficient by taking advantage of what is called preattentive perception: the brain is capable of processing specific visual information unconsciously, before it reaches higher regions of the brain. Few reduces the list of preattentive attributes originally described by Colin Ware ([War12]) to the following: [Few13, p. 80]

- Color (hue, intensity)
- Form (line length, line width, orientation, size, shape, added marks, enclosure)

- Position (2-D location)

- Motion (flicker)

Differences in line length and 2-D position are the optimal means for expressing quantitative information. Other attributes like color intensity, line width, size, orientation (when used in the right context) can also be used with this purpose but are perceived with less precision. Motion (flicker speed) is also perceived quantitatively but drives too much attention and should only be used to highlight exceptional circumstances. Shape can be used to differentiate data sets or alert icons. Line width, added marks can be used for highlighting, enclosure for both highlighting and grouping data sets together. The color palette should consist of “natural”, “soft” colors which do not irritate the sight, reserving bright and fully saturated colors for highlighting.

There is a limit of about five different expressions of a preattentive attribute the average viewer can distinguish easily, so it is not recommended to exceed that limit.

Gestalt principles, developed by German psychologists in the early 20th century, explain the human tendency to perceive groups of objects as parts of a bigger whole when presented in certain ways. The 6 Gestalt principles important in design are the following: proximity, similarity, enclosure, closure, continuity, connection[Few13, p. 87].

The principle of proximity states that objects placed close to one another on a 2-D space are perceived as belonging together (Figure 2.1). Objects that look similar to one another scattered among other, different elements are also perceived as a group (similarity principle, Figure 2.2). Surrounding a set of objects with a frame (enclosure) is another way to separate them from the rest (Figure 2.3). The principle of closure means viewers tend to see the whole picture when it is not completely visible (Figure 2.4). Objects that are aligned with one another are perceived as a whole (Figure 2.5). Objects connected to one another are also perceived as part of the same group (Figure 2.6).
2. Fundamentals

Figure 2.1: Proximity: Objects placed close to one another on a 2-D space are perceived as belonging together.

Figure 2.2: Similarity: Objects that look similar to one another scattered among other, different elements are perceived as a group.

Figure 2.3: Enclosure: Surrounding a set of objects with a frame (enclosure) is another way to separate them from the rest.
2.1 Dashboard

Figure 2.4: Closure: The principle of closure means viewers tend to see the whole picture when it is not completely visible.

Figure 2.5: Continuity: Objects that are aligned with one another are perceived as a whole.

Figure 2.6: Connection: Objects connected to one another are also perceived as part of the same group.

2.1.4 Key goals

Well-designed dashboards deliver information that is [Few13, p. 94]

• Exceptionally well organized
• Condensed, primarily in the form of summaries
• Specific to the task at hand and customized to communicate clearly to those who will use it
• Displayed using concise and often small media that communicate the information in the clearest and most direct way possible
2. Fundamentals

That is, only relevant information must be present on a dashboard and it has to be delivered directly, clearly and without unnecessary distractions. To achieve this goal, all overhead visual stimuli have to be reduced to a minimum. A useful concept here is what is called the data-ink or data-pixel ratio [Few13, p. 98] Non-data pixels (borders, grids, unnecessary colors and visual effects) should be removed or de-emphasized, while data pixels should be enhanced.

Elements on a screen are perceived with different levels of emphasis depending on their location. Probably as a result of the conventions of Western written languages, users in these regions scan the screen from left to right and from top to bottom, resulting in the configuration on Figure 2.7.

![Figure 2.7: Different regions of the screen are associated with different emphasis levels.](image)

2.1.5 Graphs

As already stated, information presented graphically can be perceived more efficiently than when presented as text. This is why using graphics whenever possible is encouraged. This chapter presents the most useful graphs for representing different kinds of information.

Bullet graphs (Figure 2.8) are an invention of Few to substitute gauges. Useful for expressing single key measures compared to a related measure, like a target, or in the context of quantitative ranges with qualitative labels.
2.1 Dashboard

Bar graphs are the best way to display discrete values along a nominal scale. They represent quantitative information more clearly than pie charts, as will be explained at the end of this section. (Figure 2.9).

Stacked bar graphs are the right choice only if we must display multiple instances of a whole and its parts, with emphasis on the whole [Few13, p. 126]. For example, in the Figure 2.10, several instances (consecutive nightly builds) of a whole (overall number of test results) and its parts (number of passed/failed/etc. tests) are displayed with the help of a stacked bar graph.
2. Fundamentals

Figure 2.10: Stacked bar graphs are the right choice only if we must display multiple instances of a whole and its parts, with emphasis on the whole.

Dot plots (Figure 2.11) are useful when rendering a quantitative scale that doesn’t begin at zero. They represent values based on position alone, not height [Few13, p. 128].

Figure 2.11: When the scale does not begin at zero, plots that encode values based on height or area can be confusing. In this cases a dot plot may be the best choice.

Line graphs can be used to emphasize the overall shape of an entire series of values: connecting individual values gives a sense of continuity. Although, they are only appropriate along an interval scale, which subdivides continuous range of
quantitative values into equal, sequential sections, not when values are discrete.

*Small multiples* (Figure 2.12), a term popularized by Edward Tufte (p.142), refers to a tabular arrangement of similar graphs. It is useful when there are too many variables for a single graph to represent the data clearly enough [Few13, p. 128]. The dashboard designed in this project uses this approach: the data is organized into six similar quadrants, each of them showing the data for one SYNECT domain (see Figure 4.1)

![Small multiples example](image)

**Figure 2.12:** Small multiples are a tabular arrangement of similar graphs. Image source: Wikipedia [Wik16a].

*Sparklines* (Figure 2.13), an invention of Edward R. Tufte, are a highly condensed form of data display. They provide a quick sense of historical context of a measure.

![Sparklines example](image)

**Figure 2.13:** Sparklines provide a quick overview over the evolution of a measure in time. Image source: Wikipedia [Wik16b].

Box plots, scatter plots, spacial maps, heat maps and tree maps are useful to represent complex statistic and are not needed for this project.
Dysfunctional graphs that should be avoided include the pie chart, since the bar graph communicates part-to-whole information more clearly: preattentive attribute of line length encodes quantitative data better than 2-D areas. In general, “area graphs”, which use 2-D space to encode quantitative values are prone to inaccurate interpretation and occlusion problems (overlapping areas). Radar graph are also not recommended, since they are less effective than bar graphs: It is more difficult to read and compare values arranged in a circle (linear scanning is more efficient).

2.1.6 Other display media

**Icons:** Simple images with a clear meaning. Most useful are those which communicate alert, up/down and on/off.
Alert icons must be only used to draw attention to exceptional circumstances, which require immediate attention. If used too often the meaning is lost. Using more than two alert levels is not recommended.
By convention, up/down icons are usually triangles pointing either way. Sometimes colors are added to differentiate them.
On/off icons are usually simple marks (like check marks, Xs or asterisks).

**Text:** Needed when precise values have to be communicated. It should only be used when necessary. Information that is needed only once or infrequently should not be shown on the dashboard but rather be put on a separate screen.

**Images:** Images add much visual complexity to the picture and take up valuable space. They should be avoided if possible.

**Drawing objects:** Used to connect/arrange pieces of information in relation to one another.

2.1.7 Critical design practices

This chapter focuses on some critical design strategies that should be considered before starting with the design process.[Few13, p. 183]

**Organize information to support its meaning and use:** The relative position of the items on the screen is important for a correct interpretation. Items belonging together should be close to each other and separated from the rest. Also, information must be organized in a way that supports meaningful comparisons.

**Maintain consistency to enable quick and accurate interpretation:** Differences draw attention. It is recommended to use the same type of display media for similar measures and avoid unnecessary variety. Only the information the dashboard communicates should change over time, not the ways of displaying it.
Put supplementary information within reach: Details do not belong on a dashboard, which is supposed to offer only a quick overview, but there should be means of accessing them when needed. Several options are available, such as pop-ups that appear on click or hovering over an item, temporarily altered states by sustained click or additional screens that open on demand.

Expose lower-level conditions: Sometimes important information gets “hidden beneath the surface” on a dashboard. A way to inform the user that an issue needs attention on a lower level should be provided.

Make the experience aesthetically pleasing: Aesthetics have an important psychological effect and contribute to usability. While superfluous decorations should be avoided on dashboards, if several aspects are kept in mind it is not difficult to achieve an aesthetically appealing design. For instance, natural, low saturated colors are pleasing and relaxing for the sight. These should dominate the view. Bright, saturated colors irritate the sight and demand attention, they should be reserved for special situations that need immediate response. The contrast between the data and the background must be high enough, so that the first stands out clearly. The bright white of the standard screen can be softened with a touch of color so that it is less irritating. Resolution, obviously, should be as high as possible to avoid forcing the eye to bring fuzzy items into focus. Content that is not properly aligned can also create discomfort. As for the text, the use of sans-serif types is preferred, for they are less affected by possible screen resolution limitations. For consistency, the same font should be used throughout the dashboard.

Prevent excessive alerts: The purpose of alerts is to draw attention to an urgent issue. When used too often this functionality gets lost. Besides, users can feel annoyed by constant “noise” of this kind.

Keep viewers in the loop: Sometimes it is important to inform users of issues even if these do not require an immediate action from their side. The purpose of a dashboard is to help users maintain “situation awareness” (see 2.1), not only to trigger response to specific problems.

Accommodate real-time monitoring. For dashboards that are updated several times a day, specific aspects, which enhance real-time monitoring, should be kept in mind:

- Content should be reduced to a minimum (the complexity a user can handle is inversely proportional to the frequency with which information changes on a dashboard)
• Means to temporarily halt updates must be provided (so that the user can take his time to examine the data before it changes again)

• Audio alerts can be helpful (to get the users attention when he is not looking at the screen)

• Time-stamping alerts is important, so that the user can know when they occurred the next time he checks the dashboard.

2.2 Contextual Design

“Contextual Design is a structured, well-defined user-centered design process that provides methods to collect data about users in the field, interpret and consolidate that data in a structured way, use the data to create and prototype product and service concepts, and iteratively test and refine those concepts with users.” [HB]

Developed by Hugh Beyer and Karen Holtzblatt, contextual design is a user centered design process which focuses on studying the users’ work practices and designing specifically to enhance these.

It is a very logical idea that observing the users’ behaviors in their work environment, questioning them about their tasks and working closely with them on the prototypes ensures that the end product will most closely fit their needs. Beyer and Holtzblatt have developed a whole methodology based on this thinking. Following strictly every step of the contextual design process as described by them seems unnecessary for such a small project, but it is only logical to apply some of their methods and principles in this scenario, since the dashboard will be used exclusively by the SYNECT team and the design process will take place entirely on the field.

Contextual design, as described by Beyer and Holtzblatt, is based on the following principles:

1. System design must support and extend users’ work practice, where “work practice” refers to the complex and detailed set of behaviors, attitudes, goals and intents that characterize a set of users in a particular environment.

2. People are unable to articulate their own work practice, due to the fact that work practice is complex and varied and people are not consciously aware of all of its details.

3. Good design requires partnership and participation with users, for a detailed understanding of their work practices and incorporating valuable feedback to the design process.

4. Good design is systemic. Contextual Design provides methods that help keep the design coherent.
5. Design depends on explicit representations: Physical representations are extremely useful during the design process. Each technique in Contextual Design has its own tangible representation that supports doing the work, capturing the result, and sharing it with others.

Beyer and Holtzblatt propose the following five models to capture the complexity of a work context from different perspectives:

- The flow model captures communication and coordination between people to accomplish work. It reveals the formal and informal work groups and communication patterns critical to doing the work. It shows how work is divided into formal and informal roles and responsibilities.

- The cultural model captures culture and policy that constrain how work is done. It shows how people are constrained and how they work around those constraints to make sure the work is done.

- The sequence model shows the detailed steps performed to accomplish each task important to the work. It shows the different strategies people use, the intents or goals that their task steps are trying to accomplish, and the problems getting in their way.

- The physical model shows the physical environment as it supports or gets in the way of the work. It shows how people organize their environments to make their work easier.

- The artifact model shows the artifacts that are created and used in doing the work. Artifacts reveal how people think about their work - the concepts they use and how they organize them to get the work done.

The authors also provide guidelines for the design process, which include group sessions with users and usability testing procedures. This project does not closely follow all steps due to resource limitations.

### 2.3 Scrum

Scrum is a family of agile software development processes based on early and continuous (incremental) prototype development around a so-called backlog (involves continuous testing and integration). The backlog is a list of items that describe changes to be made to the product, ordered by priority. These items are processed in short, two to four weeks' intervals, called sprints. A kick-off meeting takes place at the beginning of each sprint and a retrospective meeting at the end. Short daily meetings (Daily Scrum) are held to organize the work for the day.
Scrum is a team centered approach: A Scrum team usually consists of developers with similarly distributed knowledge, where theoretically every team member could take over any given task. At the kick-off meeting the team decides which backlog items can be processed in the current sprint. They are broken down into tasks small enough to be manageable by one team member in one day. Bugs are reported as soon as they arise and have to be fixed as soon as possible, for a working version of the product must be ready at the end of a sprint.

Apart from the development team two other important roles are defined in this scenario: Product owner (PO) and scrum master (SM). The PO represents the stakeholders’ interests. Usually only POs edit the backlog and decide which items are to be processed in the upcoming sprint. The SM is not involved in the planning or technical development of the product but in managing the Scrum process itself. He serves as a consultant for the team and helps solve problems between team members.
3 Context Analysis

dSPACE is one of the leading producers of engineering tools for developing and testing mechatronic control systems. With a broad portfolio and cutting-edge technology, they are much in demand as a development partner in the automotive industry, aerospace and industrial automation.

SYNECT is a data management and collaboration software tool developed by dSPACE, with a special focus on model-based development and ECU (Electronic Control Unit) testing. It helps handle models, signals, parameters, tests, test results, etc., as well as their dependencies, versions and variants, and the underlying requirements throughout the entire development process.

The team responsible for the development of this product works with the agile Scrum methodology: With a Sprint length of 2 weeks, the work is divided between four smaller teams, each in charge of one or more domains (Base, Test Management, Requirements Management, Signal and Parameter Management, Model Management and Workflow). The scrum work is organized using a series of dashboards on Visual Studio Team Foundation Server 2015 (Figures 3.1 and 3.2).

Figure 3.1: SYNECT Scrum dashboard, overview
3. Context Analysis

Each team consists of a PO and several developers and testers. The test manager is a special role outside the Scrum team. He supervises the testing process of all the teams and ensures that the product fulfills all the quality standards before a new version is released.

The dashboard is placed in a central location, where it is accessible by every member of the team. It shows the results of automated tests performed on the product: Every night a build of the product in its current state is created (the so-called nightly build) and automated tests start running on it. During the day, several so-called sparse builds are created and put to test again (only the so-called unit tests, which test individual features, run on the sparse builds) every time some major change is implemented by the developers.

3.1 Work modeling

Following the Contextual Design methodology and principles, context data has been collected observing the daily routines of the users and interviewing several members of each user group (developers, testers, POs and the test manager). The work practices are described in the models depicted below.
The cultural model (Figure 3.3) is defined by the Scrum methodology. The work of the team is organized following the Scrum guidelines very closely: With a sprint length of two weeks, a kick-off meeting takes place at the beginning of each sprint where the new items added to the backlog by the POs are discussed. Every day each team meets for a fifteen minutes long Daily Scrum where the work of the day is organized (tasks assigned to team members). A review meeting takes place at the end of each sprint where it is decided which changes to the product will be included in the shippable release.
3. Context Analysis

The flow model (Figure 3.4) shows the role the dashboard plays in the work practice of the team and the users’ relations to one another.

Four user groups with different needs can be distinguished here: developers, testers, POs and test manager. The lead PO has not been considered as a separate user group since he has no special interest in the details of the team work and usually gets the information he needs from other sources.

The dashboard is regularly checked by developers, to see if there are any failed tests, which may indicate a faulty feature that must be fixed. Testers need to know which automated tests need to be updated (when changes to the product make old tests obsolete and cause them to fail, for example).

POs want to know when any tests in their team are failing and who is responsible. Developers, testers and POs are only interested in the results of their Scrum team. The test manager, on the other hand, needs an overview over all teams at once. He has to monitor the testing process and supervise the design of the test suites.

Figure 3.5: Sequence model: Developer
3.1 Work modeling

The sequence models, describing the individual steps the two main user groups undertake to accomplish their tasks, are shown in Figures 3.5 and 3.6.

The physical model (Figure 3.7) shows the office environment. It can be seen that the dashboard occupies a central space. It is the first thing which catches the attention of someone entering the office and it can even be seen from outside, through the glass door. The coffee kitchen stays right between the door and the screen, so that the dashboard can be examined during the coffee brake.
3. Context Analysis

3.2 Current solution flaws

The current dashboard (Figure 3.8) has been in use for 4 years. It was designed for a wider range of users: several departments use slightly altered versions of it.

Figure 3.8: Photo: The current dashboard and instructions, as seen from the coffee kitchen.

Beside some general design flaws, context analysis revealed user dissatisfaction on several levels:

1. Almost every user agrees on the information overhead: too much data is presented on screen, making it difficult to focus on the important issues. This is also one of the common mistakes in dashboard design according to Few (see section 2.1.1): a dashboard should not contain too much detail.

2. The information is organized in a way that does not fit the users’ needs any more: The team has been reorganized and split into several smaller groups, the tests suites have been reorganized and partly renamed, and the number of tests has grown significantly.

3. The dashboard does not fit on one screen, requiring scrolling, which is a bad practice in dashboard design according to Few (see section 2.1.1).
4. Navigation is poorly supported: It is not intuitive, links do not lead to useful information.

5. A bad use of graphics, which fail to draw attention to the most important issues:

- Pie charts do not present quantitative information clearly enough, according to Few (see section 2.1.5).

- Stacked bar graphs representing each test suite are a good choice in theory (for representing multiple instances of a whole and its parts with an emphasis on the whole in this case multiple test suites and the results of individual tests within each of them) but the amount of them is too big to render a useful insight.

- Five colors are used to represent the different test results (Figure 3.9), which is within the perception limits for preattentive attributes, but the information they provide is low level (users only need to know whether a test is OK or not, the details of the cause of the failure do not belong on the dashboard)

![Figure 3.9: Colors of the test results on the current dashboard](image)

- Alert icons do not address current issues anymore and must be changed. Also, there are too many of them (see section 2.1.6)

6. Text items are too small, which makes it difficult to read them from a normal viewing distance. Names of test suites and number of tests cannot be seen from a standard viewing distance by the average user.

7. Complicated, unclear instructions (Figure 3.10). On Figure 3.8 it can be seen that the instructions occupy a bigger space than the screen.
3. Context Analysis

3.3 User requirements

The work modeling revealed four user groups: Product owner, developer, tester and test manager.

Following the indications on usabilitynet.org [usa], a contextual inquiry has been performed with several members of each group, rendering the following results:

Developers and product owners have similar requirements:

1. Test results have to be grouped by domain or Scrum team, not by test group anymore.

2. A binary “traffic light” icon, indicating “something is wrong” in case any test in the group did not run successfully, is the only information required for each test suite inside a domain.

3. Navigation to detailed information on the test server.

4. No distracting details on the dashboard.

Testers, most of which do not even use the current dashboard due to information saturation, had an additional request: If there are any tests which did run
successfully already but failed in the last build, they need to know which these are.

The test manager needs an overview of all test results, from all teams and over several days. He needs to know the percentage of passed tests over the total of every test suite from the current build and the evolution in time of the results.

3.4 Conclusions

The context analysis showed the central role a well designed dashboard could play enhancing the work of the SYNECT team. Several rounds of interviews with users from the different groups were performed. User dissatisfaction on several levels with the current solution was revealed, along with some general design flaws. The work practices of the users were analyzed applying Contextual Design guidelines. The gathered data, put together with the users’ specific requests, was used to come up with ideas for the new dashboard design. Users feedback on the initial prototypes helped develop the final solution, which encountered a wide acceptance among the users and is currently being implemented by a co-worker.
4 Proposed Solution

Figure 4.1 shows the final look of the dashboard prototype.

![Prototype of the proposed solution.](image)

The layout is divided into six parts, corresponding to the SYNECT domains, as requested by the users. There is a traffic-light-like icon for every test suite inside each domain. It is red if any of the tests from the suite have failed, in which case the number of failed tests is shown on the right of the name of the test suite. If all the tests in the suite run successfully, the icon is green.

When hovering over the icon of any test suite a label is displayed (Figure 4.2) corresponding to the test run the results are from. The label includes the date and time of the test run.
4. Proposed Solution

Figure 4.2: Label with date and time of the test run is shown on hovering over each test suite.

If there are any tests that had already run successfully but failed in the latest build, an alert icon (yellow triangle with an exclamation mark) is shown inside the red traffic-light icon. The hoverbox over this icon includes a list with the names of these tests (Figure 4.3).

Figure 4.3: List of newly failed tests is shown along with the label when hovering over an alert icon.

Unit tests are a suite that requires special attention: these tests are linked to single features of the product and run several times a day. Unit test results are shown in the bigger donut chart (pie chart with a hole in the center) on the left of each domain area. The number of failed unit tests is shown in red in the center of the donut hole. Although pie charts are not recommended by Few (see Section 2.1.5, page 10), this graph was chosen for this measure for several reasons:
1. Users need only approximate information on the ratio of passed/failed tests; the number of tests is very big and changing constantly. The only case that requires attention is when a vast majority of tests have failed and this can be seen clearly enough on a pie chart.

2. The circular form is consistent with the traffic light icons that represent all the other test suites.

3. This kind of graph is familiar to the users from the team dashboards in TFS. Hovering over the graph shows the label of the test run, just like with the icons of the other tests (Figure 4.4). As the label contains time-stamp information, it is also possible for the user to infer the age of the information.

![Base](image)

Figure 4.4: Label with date and time of the test run is shown on hovering over each test suite.

A click on any graph or icon that indicates a failure in the tests brings up a pop-up window to an internal site where detailed information on the corresponding test results (log) can be found.

The amount of information on the dashboard is reduced to a minimum, following Few’s guidelines (see Section 2.1.4, on page 9) and specific user requirements: Simple icons were chosen to represent each test suite, except for the unit tests. Only two, standard colors represent the two different states of a test (green: passed, red: failed). One additional alert icon is used in exceptional situations. The bright yellow in high contrast with the red background immediately draws attention to that particular test suite (see Section 2.1.6, page 13). Soft colors are used, except for the alerts, as suggested by Few, in order to avoid irritating the users. There is a minimal amount of text on the screen (see Section 2.1.6). The names of the test suites and, a bit bigger, the names of each domain, guide the user.
quickly to the right spot. A sans-serif font (Tahoma) was chosen, following Few’s suggestions, to ensure readability (see Section 2.1.7, page 13). Gestalt principles of proximity and similarity (see Section 2.1.3, page 7) were used to arrange data on the screen: the results are grouped by domain and the same display media is used for all tests suites of similar kind (unit tests vs. all the other tests).

The test manager or anyone else who needs to see an overview of all tests can open the modal window with a summary view (Figure 4.5) by clicking on the button placed in the right bottom corner of the screen. The spot of minimal emphasis of the screen, according to Few (see Section 2.1.4, on page 9) was chosen to minimize the distracting effect of a lone button.

![Figure 4.5: The summary button opens an overview of all tests.](image)

This window shows a summary of all test results, sorted by test suite and not by domain as in the main view. Horizontal stacked bar charts show the numbers of failed tests, the different colors indicating the type of failure. Small pie charts on the left of each bar show the relation between passed and failed tests. This combination of graphs was chosen because usually the number of failed tests is much smaller (up to two orders of magnitude) than the number of passed tests, so putting all of them in one graph would make it difficult to distinguish the failures. This way the user can concentrate on the problems and only notice the usually almost completely green small pie chart if most tests are failing, which would in-
dicate that there is an issue that affects the whole suite. There is another graph that shows the overall number of tests that did run every day of the current sprint. It can be seen that the portion of the bar on the top corresponding to the failed tests is very small in comparison to the green part, but since the focus of this graph is not on the failures but on the chronological development of the testing process, this is not a problem.

The prototype on Figure 4.1 was implemented in HTML/CSS. The prototyping and modeling was done using Microsoft Power Point.
5 Summary and Future Work

The goal of this project was to design a dashboard for a very specific set of users: an agile software development team at a leading high-tech company. This chapter summarizes the design process and draws conclusions on the results. A quick note on the future of this project is put at the end.

5.1 Summary

Two main sources were studied in preparation of the design process: Stephen Few’s work on dashboard design, extracted from his book “Information dashboard design: displaying data for at-a-glance monitoring” [Few13], and Beyer’s and Holtzblatt’s Contextual Design methodology, from the authors’ website [HB].

In his work, Few defines the concept of a dashboard, describes the most important issues to have in mind during the design process of such a tool, explains the essential visual perception principles and psychological aspects involved, provides a collection of the most useful graphs and display media and offers an insight of the most critical design practices. Few’s guidelines are observed carefully in this project, aiming to offer a visually pleasant and thoughtfully designed solution.

The second source [HB] provides a structured handbook of user centered design techniques and the conceptual framework for a design process centered on the users. Contextual analysis was performed on the field applying Beyer’s and Holtzblatt’s methods. Data collected through several rounds of interviews with different users and observation of their daily work practices was put together into models (Figures 3.3, 3.4, 3.5, 3.6 and 3.7), helping to draw a picture of what were the needs of the team and the problems of the currently used solution.

The context analysis revealed a variety of needs among the different team members, adding additional challenge to the task. Several user groups were established, trade-offs being necessary to meet everyones most important requests.

Since the dashboard has not been implemented yet, a proper usability test could not take place. But the HTML prototype was presented to the users in a last round of individual meetings and was met with enthusiastic approval by every participant.
5.2 Future Work

As mentioned in the previous section, a proper usability testing process should be undertaken once the dashboard is implemented and running. Since human resources have already been assigned to the task, this should be possible soon enough.

Several other teams at the company have work practices similar to the SYNECT team and are currently using slightly modified versions of the old dashboard. Many of them have shown interest in this project and will be requesting their own version of the new design. There are chances that this design or a slightly modified version of it will become the company standard for test management dashboards in the next years.
Bibliography


